Politecnico di Torino

Politecnico di Torino

Corso di Laurea Magistrale in Ingegneria del cinema e dei mezzi di comunicazione

Dipartimento di Automatica e Informatica

A.a. 2021/2022

Sessione di Laurea Luglio 2022

Speech Emotion Recognition:

riconoscimento dello stato emozionale dalle sorgenti audio simulate alla realtà educativa delle videolezioni

Relatori:

Farinetti Laura

Canale Lorenzo

Candidati:

Silvestro Federica

"Il vostro tempo è limitato, perciò non sprecatelo vivendo la vita di qualcun'altro. Non rimanete intrappolati nei dogmi, che vi porteranno a vivere secondo il pensiero di altre persone. Non lasciate che il rumore delle opinioni altrui zittisca la vostra voce interiore. E, ancora più importante, abbiate il coraggio di seguire il vostro cuore e la vostra intuizione: loro vi guideranno in qualche modo nel conoscere cosa veramente vorrete diventare. Tutto il resto è secondario."

Steve Jobs, Discorso all'Università di Stanford, 12 giugno 2005

Sommario

Nello studio condotto in questa tesi sono stati presentati ed analizzati, attraverso un excursus sia teorico che sperimentale, diversi modelli di apprendimento al fine di raggiungere come obiettivo il riconoscimento delle emozioni sul parlato; nello specifico, su audio estratti da registrazioni mp4. L'argomento della tesi è la Speech Emotion Recognition.

Sono stati ricavati e messi a confronto i risultati ottenuti dai test effettuati con svariate architetture di reti neurali, ciascuna con alla base algoritmi di apprendimento diversi.

Il lavoro di ricerca condotto, oltre a mostrare quali modelli risultano più efficienti ed efficaci nel riconoscimento delle emozioni e quali caratteristiche nella costruzione di tali modelli hanno maggiore impatto sulle loro performance, ha permesso l'applicazione di tali metodologie su videolezioni di diversi corsi universitari, tenuti al Politecnico di Torino.

Tale scopo è stato raggiunto facendo leva su meccanismi di feature extraction, al fine di estrarre i componenti utili alla predizione, e confrontarli con dati etichettati provenienti da un dataset audio di riferimento. Utilizzando le tecniche di classificazione e di confronto dei dati e sfruttando la potenza del linguaggio di programmazione Python, linguaggio ad alto livello, è stato realizzato uno script che ha permesso il raggiungimento dell'obiettivo prefissato nella tesi. Infine, per condurre questa ricerca, i dati utilizzati come riferimento per l'apprendimento delle reti provengono da dataset audio e video in lingue diverse e interpretati da attori di sesso ed età distinte. I dataset di riferimento sono EMOVO, Ravdess, Tess, Savee e Berlin.

Una volta applicate le reti neurali realizzate, aventi caratteristiche di composizione e di profondità differenti, sulle videolezioni sopracitate; per poter trarre delle conclusioni utili al fine della ricerca presentata, è stata fatta una validazione dei risultati attraverso test effettuati su agenti umani.

Il confronto dei risultati ottenuti dall'analisi umana delle registrazioni con i dati dedotti da una rete neurale ha permesso di giungere ad importanti considerazioni sull'argomento trattato.

Premessa

Negli ultimi anni molti cambiamenti si sono verificati nell'ambito scolastico. Se prima gli insegnamenti erano condotti in presenza e le interazioni tra docente e studente avvenivano in un ambiente fisico; adesso, a causa della pandemia causata dal virus SARS-CoV-2, delle sempre più iscrizioni alle università ed ai corsi di formazione online, l'interazione tra docente e studente si è nettamente indebolita e si è ridotta ad un confronto tra una persona e lo schermo di un dispositivo elettronico. Si impara da una videoregistrazione e, quindi, uno dei motivi che ha spinto alla scelta dell'argomento di tesi è stato capire quanto incide sulla qualità dell'insegnamento e dell'apprendimento l'aspetto comunicativo e interattivo dei professori. È in uno scenario così complesso, che emerge l'importanza delle analisi video e/o audio e/o testuali, che si creano nel contesto delle virtual classrooms e, più in generale, negli insegnamenti svolti online. È possibile analizzare l'attenzione dello studente attraverso il suo parlare, il riconoscimento delle espressioni facciali o l'analisi dei messaggi nella chat; allo stesso modo, lato docente è possibile condurre un'analisi sulla registrazione della lezione, andando ad esaminare il suo modo di gesticolare, di parlare e le sue espressioni facciali. Partendo dall'importanza che il mondo virtuale ha rispetto al mondo fisico, si è scelto come argomento di questa tesi l'Emotion Recognition. Nello specifico, nello studio condotto ci si è focalizzati sulla Speech Emotion Recognition, ovvero sull'analisi delle emozioni a partire dalle registrazioni di video-lezioni, messe a disposizione dai professori universitari. Dalle registrazioni sono stati estratti e analizzati gli audio, al fine di comprendere le emozioni dei professori durante le loro spiegazioni. Attraverso le reti neurali artificiali, algoritmi ispirati alla struttura e al funzionamento cerebrale, è stato condotto l'apprendimento delle emozioni. Nel lavoro presentato sono stati inseriti vari test condotti con algoritmi di apprendimento differenti, sono stati utilizzati grandi quantità di dati presi da dataset differenti per le parti di training, validation e testing dei modelli; l'obiettivo ultimo è estrapolare delle conclusioni e poter effettuare confronti su quanto ottenuto. Analisi di questo tipo potrebbero essere importanti per capire come l'attenzione di uno studente e la comprensione di un concetto possano essere influenzate dalle emozioni trasmesse dal docente, dal tono della sua voce, dalla velocità del discorso, dal ritmo e, da molte altre caratteristiche, estrapolabili da una registrazione audio. Inoltre, queste analisi potrebbero risultare utili al fine di migliorare la qualità dell'insegnamento, del training, di tutoraggio online e/o in presenza, o ancora, aumentare l'attenzione dello studente e, quindi, permettere l'acquisizione di nozioni in modo più esauriente.

Indice

Indice delle Figure

Indice delle tabelle

Introduzione

In questo primo capitolo verrà inserito lo scenario applicativo della Speech Emotion Recognition e l'importanza che riveste nella realtà quotidiana. Si passerà, poi, ad illustrare brevemente gli obiettivi della tesi redatta e presentare la sua struttura generale.

1.1 Scenario di inserimento della Speech Emotion Recognition

Negli ultimi anni l'Emotion Recognition ha acquisito un'importanza sempre maggiore grazie ai molteplici campi di applicazione, come la medicina, il marketing, la guida automatica, la social robotics, l'affective computing. Nonostante l'interpretazione delle emozioni sia molto spesso soggettiva e poco chiara, mediante le tecniche di Deep Learning sono stati recentemente raggiunti interessanti risultati.

L'emozione è l'insieme di reazioni organiche, che un individuo sperimenta in risposta a stimoli, esterni o interni, che gli consentono di adattarsi nel contesto in cui si trova e rispetto a persone, oggetti e luoghi. La parola emozione deriva dal latino *emotio*, che significa "impulso". Un'emozione può essere espressa attraverso la mimica facciale, il linguaggio del corpo ed il tono della voce. L'espressione facciale è un modo importante per trasmettere le emozioni ed è da considerarsi il segnale più potente, naturale ed universale per esprimere una condizione emotiva. Per quanto riguarda il parlato, questo è ricco di informazioni paralinguistiche e linguistiche; dove l'emozione è un'informazione paralinguistica veicolata, in parte, dal discorso. Il riconoscimento vocale e, più nello specifico, il riconoscimento delle emozioni a partire dal parlato, è un campo di studio attivo e con ruolo preponderante quando si prendono in considerazione la comunicazione uomo-macchina, l'apprendimento automatico e, in generale, l'ambito dell'intelligenza artificiale.

Nell'intera esperienza umana e nel processo decisionale un ruolo cruciale giocano le emozioni. Con lo sviluppo tecnologico, l'interazione uomo-macchina si è spinta oltre la logica dei calcoli, e con l'aumento della diffusione dei dispositivi tecnologici e di realtà virtuali, la comprensione delle emozioni di un utente è utile per la realizzazione di interfacce più chiare e naturali. Con il termine Speech Emotion Recognition indichiamo un insieme di metodi che elaborano segnali vocali per rilevarne le emozioni incorporate. La sfida è riconoscere l'emozione dell'oratore tramite una classificazione fatta da un automatic emotion recognizer, sistema realizzato ad hoc in grado di apprendere, ovvero generalizzare una regola ed applicarla al fine di predire nel modo corretto l'emozione reale di partenza.

1.2 Obiettivi

L'obiettivo che ci si è proposti di raggiungere riguarda lo sviluppo di un modello di apprendimento, che permetta di comprendere l'emozione di un parlatore, a partire da una registrazione audio, sulla base dell'analisi del contenuto informativo della registrazione di riferimento. Sfruttando la potenza di Python è stato realizzato un codice in cui, in ordine di inserimento dei blocchi di codice, avviene:

- l'isolamento della componente audio da video-registrazioni.
- l'estrazione delle features utili.
- la realizzazione di una rete neurale con tutte le caratteristiche ad essa associate, al fine di avere un apprendimento il più efficiente possibile.
- l'utilizzo di dati per il training, la validation ed il testing a partire da dataset specifici.
- la predizione, ovvero il confronto delle predizioni con i dati reali e l'analisi dei risultati.

1.3 Struttura della tesi

La tesi è organizzata nel seguente modo:

Capitolo 1: include una breve introduzione all'argomento, gli obiettivi ed un paragrafo illustrativo della struttura della tesi.

Capitolo 2: contiene un breve excursus sulla storia del SER ed una parte teorica ed illustrativa relativa ai concetti fondamentali su cui si fonda tutta la tesi. Riscontriamo spiegazioni teoriche sui dataset, sulla feature extraction, sulla generazione del parlato, sui vari modelli di apprendimento e sui concetti base del Deep Learning, sui vari modelli di reti neurali e le caratteristiche ad essi associati, sulla fase di apprendimento e di predizione, sulle metriche da tenere in considerazione quando il riconoscimento termina. Si elencano i risultati ottenuti con le varie configurazioni di sistema.

Capitolo 3: sono riportati tutti i test svolti con i relativi risultati. Inoltre, è svolta un'analisi dei parametri più influenti nelle predizioni e si tiene traccia delle modifiche effettuate e dei cambiamenti, che tali modifiche hanno comportato.

Capitolo 4: è presentato ed illustrato il modello realizzato, al fine di raggiungere l'obiettivo della tesi con il risultato migliore. Inoltre, contiene considerazioni su quanto riscontrato nei test.

Capitolo 5: contiene le conclusioni, un riassunto degli scopi della ricerca presentata, presenta i risultati ottenuti accompagnati dalle rispettive valutazioni e illustra le prospettive future.

Background teorico

Il secondo capitolo ha come obiettivo primario quello di fornire i concetti fondamentali del riconoscimento delle emozioni dai modelli di apprendimento. Nella prima parte ci si è focalizzati sulla voce e sull'emozione, presentando le proprietà che rispettivamente le caratterizzano. Nella seconda parte del capitolo si è passati ad analizzare le componenti fondamentali per attuare l'apprendimento emozionale di una rete neurale, ovvero si è parlato di dataset, feature extraction, model fit, delle metriche e parametri usati ed è stato inserito un excursus relativo alle varie metodologie usate per l'apprendimento emozionale automatico.

2.1 Voce ed emozione

La voce rappresenta lo strumento per eccellenza della comunicazione, fondamentale nelle relazioni umane, nell'espressione artistica e parte integrante di ogni cultura nel mondo. La voce permette di esprimere sia a livello verbale che para-verbale pensieri, opinioni e, soprattutto, stati d'animo ed emozioni. Indipendentemente dalle parole pronunciate e dal loro significato, il flusso comunicativo fornisce molteplici informazioni. Secondo studi condotti sull'espressione vocale delle emozioni, si è visto che ogni espressione risulta caratterizzata da precisi indicatori vocali e che gli ascoltatori sono in grado di riconoscere uno stato emotivo basandosi su questi. La voce risulta influenzata dai vissuti emotivi, a tal punto da modularsi a seconda delle situazioni. Elementi da attenzionare sono:

- il tono, legato alle corde vocali, identifica l'intonazione. Un tono basso o alto può comunicare a che livello sociale si appartiene, essendo influenzato da fattori fisiologici e dal contesto.
- la frequenza, che comunica la formalità delle situazioni sulla base delle sue variazioni.
- il ritmo, che è scandito da pause e dalla vocalizzazione, ma anche dalla velocità con cui un discorso è articolato per il numero di sillabe pronunciate nell'unità di tempo, comunica autorevolezza sia al discorso pronunciato che al parlante.
- le alterazioni nella comunicazione in relazione agli stati emotivi del soggetto. La voce di chi parla è influenzata dalle emozioni che prova nel momento specifico; ma, al tempo stesso, può influenzare le sue emozioni.

A seguito di numerosi studi svolti si è potuto concludere che non è importante il significato delle parole usate in un discorso, in quanto la voce da sola è in grado di trasmettere le diverse sfumature emozionali.

La soggettività e la difficoltà del riconoscimento delle emozioni a partire da una frase sono state dimostrate dalla ricerca di Luigi Anolli e Rita Ciceri. In questa ricerca è stato richiesto il riconoscimento dell'emozione, caratterizzante un audio, a duecento persone ed il 65% di loro è riuscito ad individuare ciascuna emozione senza particolari difficoltà. Le emozioni più

riconosciute sono state quelle negative; il 77% ha individuato la collera, il 73% la paura, il 70% la tristezza, mentre la tenerezza è stata individuata solo dal 40%.

Secondo tale studio i principali profili emozionali che la voce può rappresentare sono: collera, gioia, paura, tristezza, disprezzo e tenerezza.

- La collera è un'emozione espressa con voce tesa e piena, la persona usa un tono alto e nell'esprimersi non ha pause. L'intensità dei suoni risulta forte.
- La gioia ha un tono ed un'intensità elevata, la voce di chi parla è ampia. La risonanza è bilanciata.
- La paura è espressa con un tono acuto ed una voce stretta e tremula.
- La tristezza ha una cadenza ed un ritmo lento, il tono risulta basso e l'intensità moderata. La voce dell'interlocutore si presenta rilassata ma, anche, stretta.
- Il disprezzo ha come caratteristica principale la segmentazione delle sillabe. Si riconosce per una velocità del parlato normale.
- La tenerezza è espressa con voce ampia e distesa.

Tutti i concetti presentati nei paragrafi successivi di questo capitolo costituiscono la base teorica su cui si fonda lo Speech Emotion Recognition. Tali nozioni trovano la loro applicazione nei test condotti e presentati nel Capitolo 3 e nel Capitolo 4 di questa tesi. Inoltre, si è avuto cura nel presentare le nozioni nell'ordine in cui compariranno negli script di interesse.

2.2 Storia del SER

I primi studi relativi alla Speech Emotion Recognition cercavano di dimostrare la correlazione tra i parametri acustici di un discorso e le emozioni. In queste analisi vari parametri di basso livello, o gruppi di parametri, venivano analizzati per dimostrarne la correlazione con le emozioni dell'interlocutore. In generale, sono due i principali metodi per condurre il SER: metodo che utilizza i tradizionali classificatori e metodo che sfrutta gli algoritmi di Deep Learning.

Per molti anni, la tendenza è stata quella di applicare features ingegneristiche a descrittori spettrali di basso livello, estrapolati da audio, e passarle a classificatori tradizionali; come SVM (Support Vector Machine), regressioni logistiche o alberi decisionali. I descrittori analizzati erano la frequenza fondamentale e le frequenze formanti, jiiter, shimmer, energia spettrale e velocità del parlato; si è dimostrato che tutti i descrittori prima elencati sono correlati all'intensità emotiva ed ai processi emotivi. Ottimi risultati SER sono stati, poi, ottenuti attraverso l'utilizzo di parametri più complessi; parametri come i coefficienti cepstrali della frequenza di Mel (MFFC), il roll-off spettrale, le caratteristiche di Teager Energy Operator (TEO), gli spettrogrammi e le caratteristiche della forma d'onda glottale. In questo scenario primordiale, un esempio degno di nota è il lavoro di Ancilin e Milton che hanno

studiato un metodo per ottenere coefficienti cepstrali di frequenza Mel, calcolando lo spettro di magnitudo, invece dello spettro di energia.

Nella loro ricerca i coefficienti di magnitudo della frequenza di Mel e altre tre features spettrali, ovvero i coefficienti cepstrali della frequenza di Mel, i coefficienti di potenza della frequenza log ed i coefficienti cepstrali di predizione lineare sono stati testati con Berlin, Ravdess, Savee, EMOVO, eNTERFACE ed Urdu su classificatori multiclasse di tipo SVM. L'accuratezza ottenuta dai loro test era di 81.50% per Berlin, 64.31% per Ravdess, 75.63% per Savee, 73.30% per EMOVO, 56.41% per eNTERFACE and 95.25% per Urdu. Ancilin e Milton riuscirono a dimostrare che i coefficienti di magnitudo della frequenza di Mel sono le migliori features per l'identificazione delle emozioni nel parlato.

Oggi, la maggior parte delle pubblicazioni vede l'impiego dei modelli di Deep Learning, questi classificatori sono reti neurali capaci di processare i descrittori elencati nel paragrafo precedente o la registrazione audio stessa. Tra gli esempi di questo nuovo approccio possiamo citare lo studio di Singh et al, che suggerisce l'uso della prosodia, delle informazioni spettrali e della qualità della voce, per addestrare un classificatore DNN, raggiungendo una precisione dell'81,2%. Ruolo rilevante occupano, anche, gli studi che vedono l'impiego di reti CNN, MLP o LSTM per la risoluzione del riconoscimento delle emozioni su RAVDESS, utilizzando spettrogrammi o caratteristiche, elaborate in precedenza; tali studi hanno ottenuto, rispettivamente, precisioni dell'80%, 96,1% e 81%. Infine, alcuni studi si concentrano sui vantaggi del trasferimento dell'apprendimento per mettere a punto modelli pre-addestrati, piuttosto che estrarre caratteristiche in tempo reale. A questo proposito, DeepSpectrum, PANN e Hugging Face sono librerie che contengono modelli pre-addestrati su audio, immagini e/o testo.

2.3 Databases

Il riconoscimento delle emozioni, come ogni altra attività di apprendimento, necessita l'utilizzo di un set di addestramento di campioni. Il processo di realizzazione di un set di dati richiede che i campioni vengano etichettati manualmente da agenti umani; il limite è che persone diverse percepiscono le emozioni in modo differente. Questo limite comporta che, per l'operazione di etichettamento, i campioni vengano sottoposti a persone diverse e che si abbia a disposizione un sistema per scegliere con sicurezza tra le etichette disponibili quelle ritenute maggiormente valide.

Esistono tre tipi diversi di database progettati per il riconoscimento delle emozioni a partire dal parlato:

- dataset simulati, creati a partire da relatori addestrati, che leggono lo stesso testo con emozioni diverse. Dataset di questo tipo sono EMO-DB, DES, RAVDESS, TESS. La varietà di emozioni contenuta è molto ampia.
- dataset seminaturali, creati da performance di attori a cui viene richiesto di leggere uno scenario contente emozioni diverse. Dataset di questo tipo sono IEMOCAP,

Belfast. Nonostante riproducano discorsi reali, sono emozioni create artificialmente. La varietà di emozioni è limitata rispetto ai dataset simulati.

- dataset naturali, raccolgono campioni estratti da programmi TV, video Youtube, call center ed etichettati da agenti umani. L'utilizzo di questi dataset è molto complesso, a causa della continuità delle emozioni, alla variazione dinamica dell'emozione durante il discorso registrato ed a causa di registrazioni non "pulite", affette da rumori di fondo.

Se i primi dataset utilizzati nell'ambito del SER comprendevano un numero limitato di campioni con un numero limitato di attori, quelli recenti presentano audio di emozioni molto vario e campioni registrati da attori con caratteristiche differenti.

Nei paragrafi successivi verranno presentate ed analizzate le caratteristiche dei dataset di maggiore utilizzo nell'ambito della Speech Emotion Recognition.

2.3.1 RAVDESS

RAVDESS, The Ryerson Audio-Visual Database of Emotional Speech and Song, è un dataset contenente 7356 file, dove 1500 sono file solamente audio. Si tratta di brevi registrazioni di 8 emozioni, interpretate da 24 attori diversi, nello specifico 12 uomini e 12 donne. Le emozioni sono: 1 = neutrale, 2 = calmo, 3 = gioia, 4 = tristezza, 5 = rabbia, 6 = paural, 7 = disgusto, 8 = sorpresa. Ogni audio è rinominato in modo da identificare sesso ed emozione rappresentata. La caratteristica più rilevante di RAVDESS è che ogni emozione è eseguita con due diverse intensità e con voce sia cantata che normale.

2.3.2 TESS

TESS, Toronto emotional speech set, è un dataset costituito da audio di alta qualità interpretati da sole donne. Negli audio le emozioni (rabbia, disgusto, paura, gioia, sorpresa, tristezza, neutralità) sono interpretate da due attrici. I file audio sono in totale 2800 wav. Una particolarità di questo database è che nasce per analizzare l'effetto dell'età sulla capacità di riconoscimento delle emozioni; infatti, le due attrici in questione hanno 64 e 26 anni. Le emozioni in questo dataset sono: rabbia, sorpresa, disgusto, gioia, tristezza, paura, neutrale. Alle due attrici è stato chiesto di interpretare le sette emozioni, prima elencate, per 200 frasi; i dati sono stati fatti poi etichettare a studenti universitari e solo le frasi con un affidamento del 66% sono state selezionate ed inserite nel dataset.

2.3.3 SAVEE

Il dataset SAVEE, Surrey Audio-Visual Expressed Emotion, contiene circa 500 registrazioni audio con protagonisti 4 attori maschi. Le emozioni contenute sono 7, come per i dataset analizzati nei paragrafi precedenti; sono 480 le espressioni in inglese britannico presenti. Le frasi sono state scelte dal corpus TIMIT standard e bilanciate foneticamente per ciascuna emozione. I dati sono stati registrati in un laboratorio di supporti visivi con apparecchiature

audiovisive di alta qualità, elaborati ed etichettati. Per verificare la qualità della performance, le registrazioni sono state valutate da 10 soggetti in condizioni audio, visive e audiovisive.

2.3.4 EMOVO

Emovo, rappresenta il primo database applicabile alla lingua italiana. È stato realizzato mettendo insieme le voci di 6 attori, a cui è stato chiesto di recitare 14 frasi, simulando 7 diverse condizioni. Le registrazioni sono state effettuate con apparecchiature professionali nei laboratori della Fondazione Ugo Bordoni. Una trattazione completa su EMOVO è inserita nel Capitolo 4, dove verranno approfondite le sue caratteristiche e presentate le sue applicazioni nel campo della Speech Emotion Recognition.

2.3.5 EMO-DB

Il Berlin Database of Emotional Speech, noto come EMO-DB, è uno dei dataset più vasti tra quelli usati per lo Speech Emotion Recognition. Il database è stato creato dall'Istituto di Scienze della Comunicazione, Università Tecnica, Berlino e contiene 535 enunciati. È composto da 10 frasi in tedesco; nello specifico, 5 frasi brevi e 5 frasi lunghe, recitate da 10 attori differenti. Per la creazione del dataset 5 femmine e 5 maschi hanno simulato 7 diverse emozioni, neutrale, rabbia, paura, gioia, tristezza, disgusto, noia. Il dataset contiene 700 samples. I dati sono stati registrati ad una frequenza di campionamento di 48 kHz e, quindi, sotto-campionati a 16 kHz. Per l'estrazione con una maggiore precisione della qualità prosodica e vocale, oltre che le voci, sono stati registrati anche gli elettroglottogrammi.

2.4 Feature Extraction

Passaggio successivo alla scelta e all'esplorazione dei dataset è l'estrazione delle features dai file audio, indispensabile per permettere al modello di apprendere. Tracciando la forma d'onda ed uno spettrogramma si ottiene una visualizzazione dei file audio. La forma d'onda è la rappresentazione di quante e quali frequenze sono presenti nel suono analizzato, e con quale ampiezza. In definitiva, è una rappresentazione grafica delle frequenze di un suono. Lo spettrogramma, invece, è la rappresentazione grafica dell'intensità, ovvero lo spettro delle frequenze, di un suono in funzione del tempo.

A partire dalla forma d'onda e dallo spettrogramma, calcolati da un audio, è possibile stilare una scala di valutazione:

- Gioia: velocità alta, contorni netti, poche armoniche, moderata variazione di estensione. Ampia variazione di tonalità e velocità.

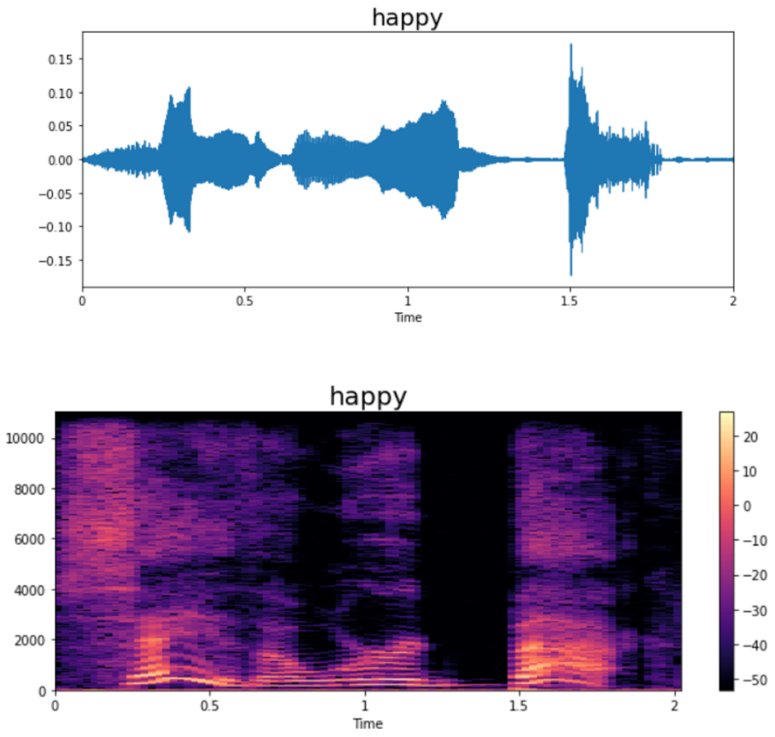
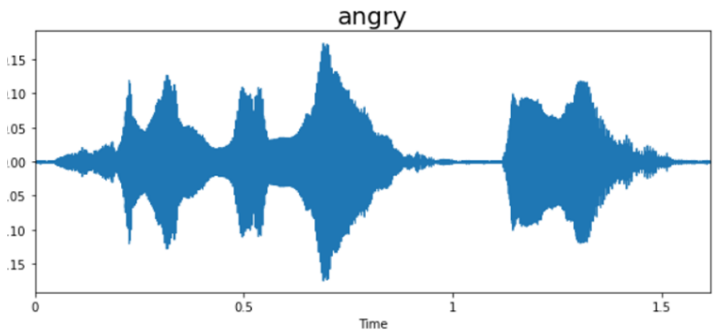


Figura 1 - Forma d'onda e spettrogramma gioia

- Rabbia, alta velocità, tonalità acuta, contorni arrotondati, contorno di tonalità ascendente. Ampia variazione di estensione e numerose armoniche.



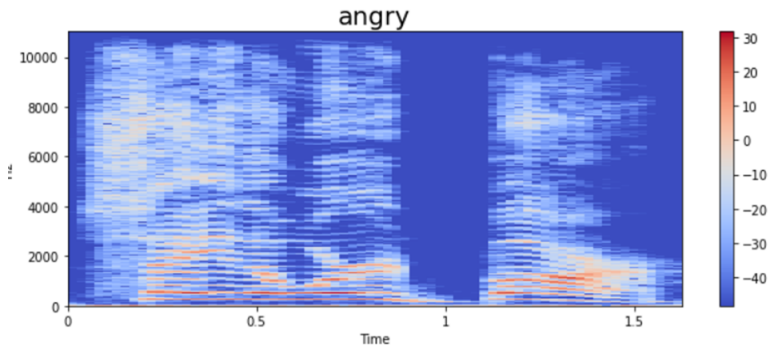


Figura 2 - Forma d'onda e spettrogramma rabbia

- Disgusto, armoniche numerose, scarsa variazione di tonalità, contorni arrotondati, bassa velocità.

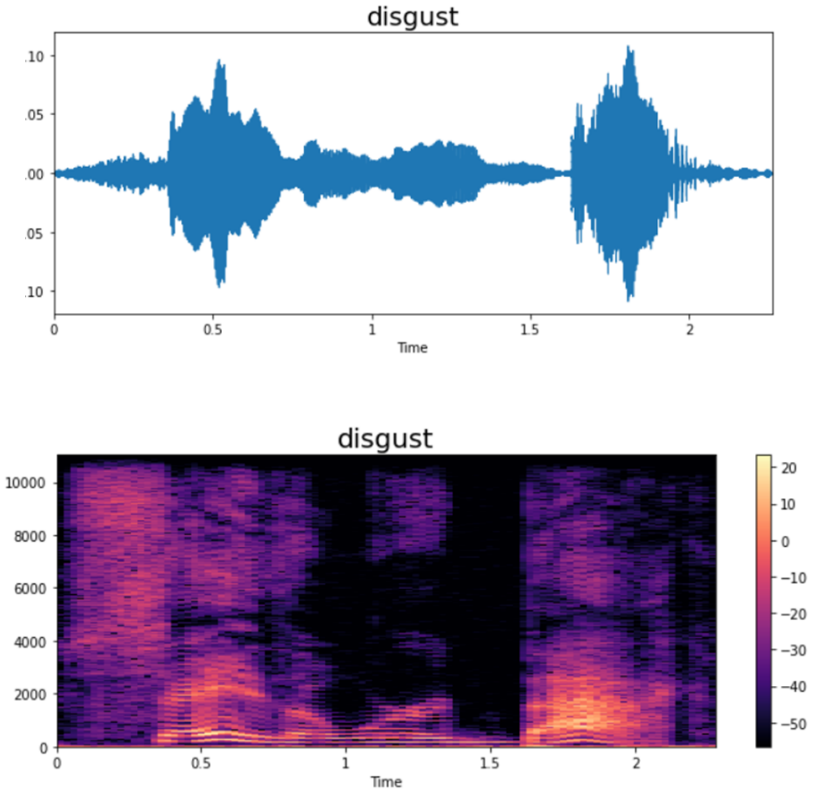


Figura 3 - Forma d'onda e spettrogramma disgusto

- Paura, contorno di tonalità ascendente, sequenza rapida, tonalità acuta, contorni arrotondati, scarsa variazione di tonalità. Alta velocità e numerose armoniche.

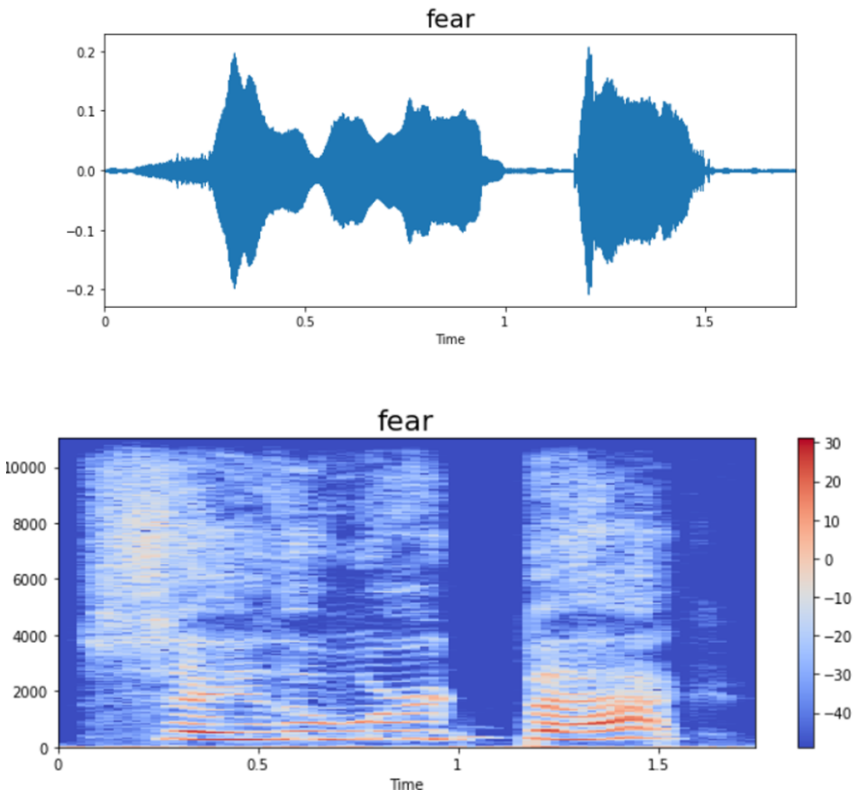
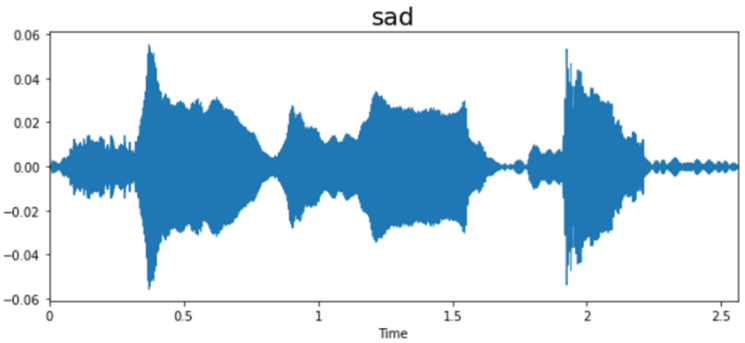


Figura 4 - Forma d'onda e spettrogramma paura

- Tristezza, velocità bassa, tonalità profonda, poche armoniche, contorni arrotondati, contorno di tonalità discendente.



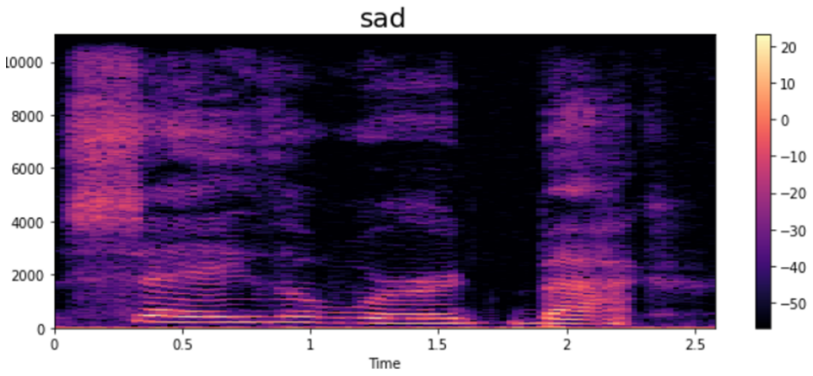


Figura 5 - Forma d'onda e spettrogramma tristezza

- Sorpresa, velocità alta, tonalità acuta, contorno di tonalità ascendente, contorni netti, ampia variazione di tonalità.

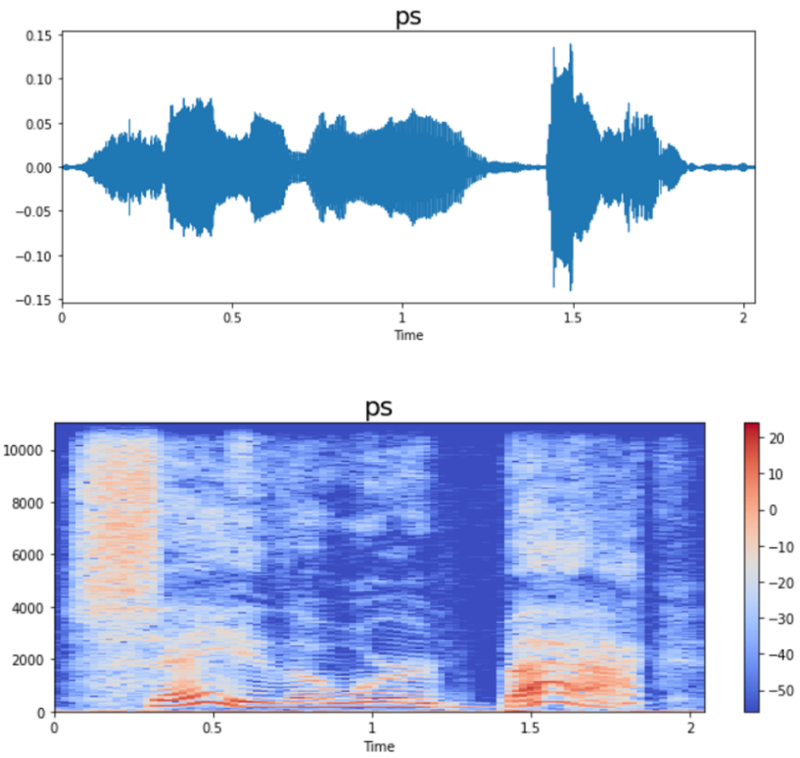


Figura 6 - Forma d'onda e spettrogramma sorpresa

- Neutrale, contorni arrotondati, velocità bassa. Bassa variazione di tonalità.

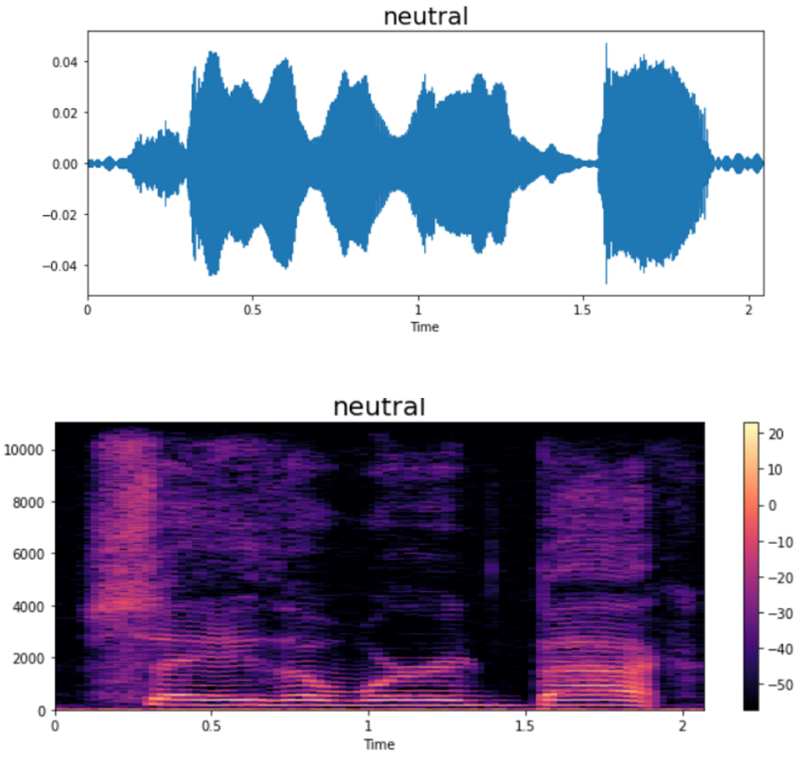


Figura 7 - Forma d'onda e spettrogramma emozione neutra

2.4.1 Generazione del parlato

Il parlato è prodotto dal tratto vocale, un sistema complesso costituito da diversi elementi che, in base a come si modella, produce suoni differenti. In termini di elaborazione del segnale digitale possiamo pensare al tratto vocale come ad un filtro. Tutto ha inizio dagli impulsi glottali, che passano attraverso il tratto vocale; quest'ultimo, filtrando il segnale iniziale, permette la creazione del segnale vocale. Gli impulsi glottali sono da vedersi come segnale acuto e rumoroso, mentre a seconda di come si modella il tratto vocale avremo un segnale vocale di uscita diverso. L'impulso glottale trasporta informazioni sull'altezza o sull'alta frequenza, quindi i fonemi e le diverse consonanti prodotte. Partendo dal segnale vocale e smussandolo si ottiene quello che viene definito inviluppo spettrale; ciò che l'uomo percepisce è ben mostrato dai picchi dell'inviluppo spettrale. I picchi sono chiamati formanti e sono responsabili dell'identità dei suoni. L'involucro spettrale rappresenta la risposta in frequenza del tratto vocale.

Il segnale vocale iniziale può essere scomposto in due parti, una contiene informazioni sulle caratteristiche spettrali, mentre l'altra è relativa all'impulso glottale. Il discorso può essere interpretato come una convoluzione della risposta in frequenza del tratto vocale con l'impulso glottale nel dominio del tempo, quindi, ci spostiamo nel dominio della frequenza.

Quando lavoriamo con il segnale vocale non abbiamo la risposta in frequenza del tratto vocale separata dall'impulso glottale; l'obiettivo è separare il segnale vocale iniziale nelle due componenti. In termini di riconoscimento vocale non siamo molto interessati all'impulso glottale, siamo interessati all'identità del suono (ai fonemi, ai tempi, alle formanti). Quindi, vogliamo una serie di funzionalità che ci consentano di lavorare solo con la parte dell'inviluppo spettrale. Partiamo dal segnale vocale e vogliamo isolare la componente di risposta in frequenza della traccia vocale.

2.4.2 MFC e MFCC

Visto che il riconoscimento delle emozioni a partire dagli audio di dataset è un compito di apprendimento supervisionato, come input non si possono utilizzare segnali audio grezzi, ma è necessario estrarne le caratteristiche per ottenere prestazioni migliori. Per l'estrazione delle caratteristiche degli audio si utilizza la libreria Librosa, nello specifico, la tecnica più utilizzata è quella che sfrutta gli MFCC. Partendo dal presupposto che qualsiasi suono generato dall'uomo è definito dalla forma del suo tratto vocale (inclusi lingua, denti, ecc…) è possibile affermare che l'inviluppo dello spettro di potenza temporale del segnale vocale è rappresentativo del tratto vocale; e a rappresentare accuratamente questo inviluppo è, proprio, il concetto di MFCC.

La rappresentazione MFC è stata inventata e sviluppata a partire dagli anni '80, con questa denominazione, mel-frequency cepstrum, si indica una rappresentazione dello spettro della potenza a breve termine di un suono, basato su una trasformata del coseno lineare di uno spettro di potenza logaritmica su una scala di frequenza Mel non lineare.

Gli MFCC si ottengono da uno "spettro di uno spettro" non lineare, ovvero da una rappresentazione cepstrale, sono coefficienti cepstrali della frequenza di Mel (MFCC), una scala che mette in relazione la frequenza percepita di un tono con la frequenza effettiva misurata, e costituiscono nel loro insieme un MFC. Generalmente, i primi 13 coefficienti (le dimensioni inferiori) di MFCC sono presi come caratteristiche, in quanto rappresentano l'inviluppo degli spettri. E le dimensioni superiori scartate esprimono i dettagli spettrali. Possiamo riconoscere i fonemi tramite MFCC.

Nella figura sottostante è illustrato l'algoritmo MFCC con tutte le fasi che lo costituiscono.

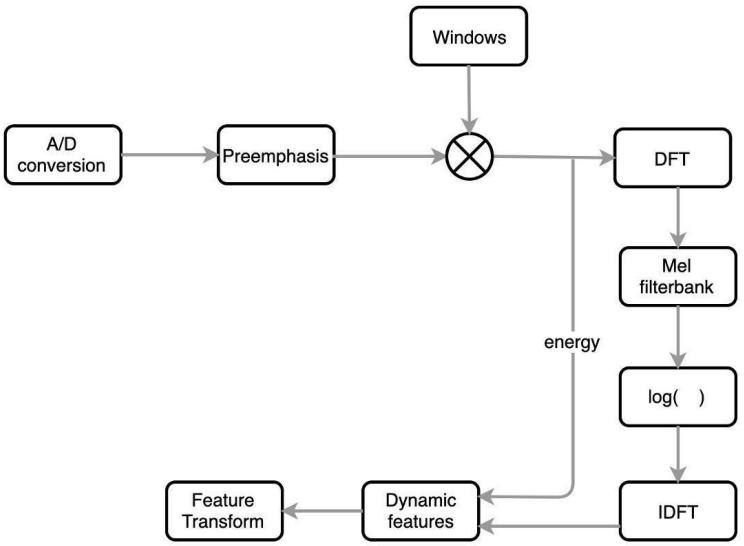


Figura 8 - Algoritmo MFCC [1]

Il primo step è la conversione del segnale audio da analogico a digitale, la frequenza di campionamento spesso utilizzata è di 8 o 16kHz.

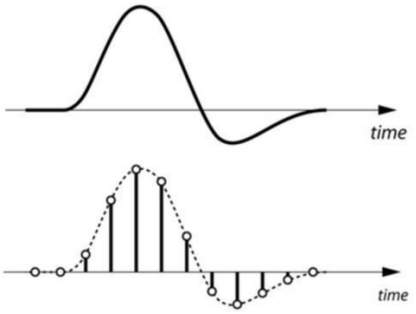


Figura 9 - Conversione segnale da analogico a digitale [1]

Per i segmenti sonori, c'è più energia alle basse frequenze; aumentare l'energia alle alte frequenze rende le informazioni delle formanti superiori più disponibili. La fase della pre-enfasi aumenta la quantità dell'energia nelle alte frequenze attraverso l'utilizzo di un filtro. Le caratteristiche di un segnale vocale rimangono stazionarie in un intervallo di tempo sufficientemente corto, per questo motivo i segnali vengono elaborati in intervalli di tempo ridotti. Altra operazione importante è eliminare le discontinuità alle estremità di ogni frame, non si può tagliare il segnale ai bordi in modo netto, perché questo causerebbe una caduta improvvisa di ampiezza e creerebbe rumore, visibile alle alte frequenze. Per tagliare l'audio si usa quindi la finestra di Hamming o di Hanning, in questo modo l'ampiezza diminuisce gradualmente al bordo di un fotogramma.

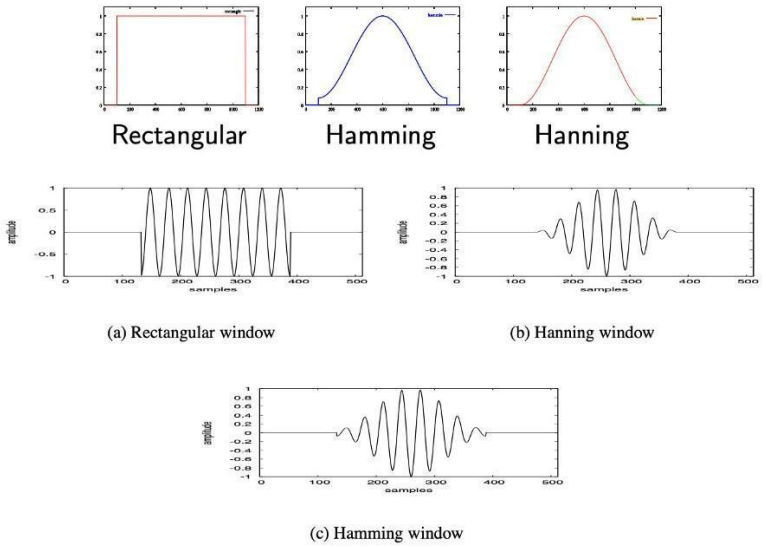


Figura 10 – Windowing [1]

Viene effettuato, poi, il passaggio dal dominio del tempo a quello della frequenza, applicando la trasformata di Fourier discreta (DFT). Grazie a questa conversione l'analisi è semplificata, viene calcolato così lo spettro del segnale ed è possibile sapere quanto una componente frequenza è presente nel segnale di origine. Nell'estrazione delle caratteristiche del segnale, l'output della DFT viene elevato al quadrato, in modo da ricavare lo spettro di potenza, a questo viene applicato il banco di filtri triangolari su scala Mel, in modo da imitare ciò che l'essere umano percepisce.

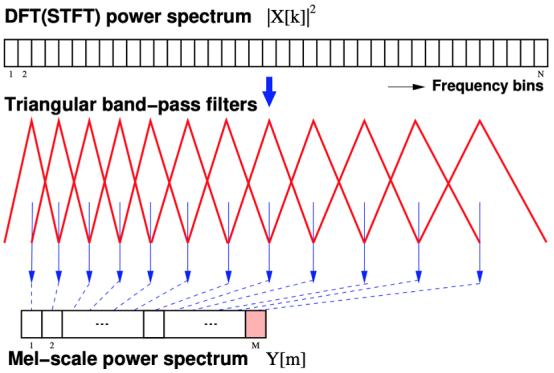


Figura 11 - Operazioni per l'ottenimento dello spettro di potenza in scala Mel [1]

Le nostre orecchie hanno una risoluzione maggiore ad una frequenza inferiore rispetto a quella che si ha a frequenze più alte; per questo motivo si mappa la frequenza attuale nella scala di Mel, che è una scala di percezione dell'altezza (pitch) di un suono, attraverso la formula:

$$mel(f) = 1127 \ln(1 + \frac{f}{700})$$

Gli esseri umani sono meno sensibili ai piccoli cambiamenti di energia ad alta energia rispetto ai piccoli cambiamenti ad un basso livello di energia, questa proprietà è simile a quella del logaritmo. Quindi, l'altro passaggio consiste nell'applicare un logaritmo ad uno spettro di potenza; a tutte le ampiezze si applica un logaritmo in modo da avere i decibel, si ricava lo spettro di potenza logaritmica.
Infine, viene applicata una trasformata inversa coseno, in questo modo si ottiene lo spettro di uno spettro, denominato cepstrum. Quest'ultimo ha un asse con una pseudo frequenza, chiamata Quefrency, mentre sull'asse delle y è possibile leggere la magnitudine assoluta. Una volta applicata la trasformazione presentata sopra, si ottengono un numero di coefficienti, che sono gli MFFC. Nel cepstrum sono presenti dei picchi, che mostrano quanto siano presenti le diverse quefrency nello spettro della potenza logaritmica; il picco enorme è il primo rhamonic, questa è la quefrecy che è associata alla frequenza fondamentale del segnale originale.
Riassumendo per calcolare gli MFCC servono i seguenti passaggi:

- Calcolare la trasformata di Fourier di (un estratto in finestra di) un segnale.
- Mappare le potenze dello spettro ottenuto sulla scala Mel, utilizzando finestre triangolari sovrapposte o, in alternativa, finestre coseno sovrapposte.
- Prendere la trasformata inversa discreta del coseno dell'elenco delle potenze mel log.
- Gli MFCC sono le ampiezze dello spettro risultante.

2.5 Metodi per l'Emotion Recognition

Il riconoscimento delle emozioni, a partire dal parlato, può essere effettuato utilizzando metodi e algoritmi differenti; ognuno dei quali risolve il problema da un'angolazione specifica e presenta vantaggi e svantaggi. Come anticipato nel paragrafo contenente la storia del SER, storicamente il metodo più diffuso in questo ambito era quello basato sull'utilizzo di algoritmi di Machine Learning classici, come HMM e SVM. Successivamente, vista la grande rilevanza che ha acquisito questo argomento di studio, a seguito di vari esperimenti e ricerche condotte, ci si è mossi verso l'utilizzo di metodi di Deep Learning. Nei paragrafi successivi verranno analizzate le varie metodologie su cui fonda la sua esistenza l'Emotion Recognition.

2.5.1 HMM e SVM

Nelle prime ricerche condotte per il riconoscimento delle emozioni a partire da un audio, un ruolo cruciale hanno occupato HMM e SVM.

I modelli nascosti di Markov per la prima volta sono stati descritti negli studi statistici di Leonard E. Baum degli anni ‘60. Una delle prime applicazioni è stata il riconoscimento della parola negli anni ‘70. I modelli HMM permettono di classificare anche le variazioni minime in un segnale audio, in particolar modo si parla di riconoscimento dello schema temporale dei discorsi parlati. Si tratta di un approccio statistico; applicando tale approccio, una sequenza viene modellata come output di un processo che procede attraverso una serie di stati, che sono "nascosti" all'osservatore. Ciascuno di questi stati nascosti emette un simbolo, che rappresenta un'unità elementare dei dati modellati.

Quando l’obiettivo è la classificazione o la regressione, un algoritmo di apprendimento automatico supervisionato usato è quello SVM, ovvero Support Vector Machines. Nell'algoritmo SVM, la classificazione è eseguita costruendo un iperpiano N-dimensionale in cui ogni elemento di dati è mappato come punto nello spazio; in tale spazio i dati sono divisi in categorie. I dati possono essere ridimensionati prima di essere applicati ad un classificatore SVM, per evitare attributi in intervalli numerici maggiori durante l'elaborazione. Il ridimensionamento serve, anche, allo scopo di evitare alcune difficoltà numeriche durante il calcolo. L'idea principale in SVM è l'utilizzo di una funzione del kernel per trasformare l'input originale, impostato in uno spazio di funzionalità ad alta dimensione; viene, così, permessa una classificazione ottimale nel nuovo pazio di funzionalità. Il vantaggio dell'utilizzo del kernel RBF, Radial Basis Function, è che limita i dati di addestramento, mantenendoli entro limiti specifici. SVM è stato applicato al database delle emozioni Berlin e sono state estratte le caratteristiche MFCC e MEDC dai file vocali in formato .wav. Utilizzando la funzione del kernel polinomiale e RBF, i risultati di LIBSVM erano rispettivamente del 93,75% e del 96,25%.

2.5.2 Neural Network e Deep Learning

Nonostante le reti neurali artificiali abbiano visto il loro sviluppo negli anni '80, il loro utilizzo era ridotto a causa della mancanza di velocità di processo e della scarsa memoria. Queste architetture erano poco profonde, uno o due livelli, quindi i risultati ottenuti erano limitati. Oggi, grazie al miglioramento della memoria, della CPU e della potenza della GPU dei computer, è possibile progettare e testare architetture più sofisticate e più profonde.

Le reti neurali più comuni sono: rete convoluzionale (CNN, Convolutional Neural Network), rete neurale deconvoluzionale (DNN, Deconvolutional Neural Network), reti LSTM (Long short-term memory), rete generativa avversaria (GAN, Generative Adversarial Network), rete neurale ricorrente (RNN, Recurrent Neural Network), convertitori.

Una sottocategoria del Machine Learning è rappresentata dal Deep Learning, la cui traduzione letterale sarebbe "apprendimento profondo". Alla base del Deep Learning troviamo algoritmi ispirati sia alla struttura che alle funzionalità cerebrali, si parla di reti neurali artificiali. Emulando il modo in cui gli esseri umani ragionano per ottenere un certo tipo di conoscenza, ci si riferisce ad un modo per automatizzare l'analisi predittiva. Dal punto di vista scientifico l'apprendimento viene portato avanti tramite dati dedotti grazie all'utilizzo di algoritmi, prevalentemente di calcolo statistico.

Il Deep Learning simula il processo di apprendimento del cervello umano attraverso sistemi artificiali per insegnare alle macchine ad apprendere in modo indipendente, facendolo come il cervello farebbe, ovvero su più livelli. Il modello suddetto si basa sul paradigma biologico della mente proposto dai premi Nobel Hubel e Wiesel nel 1959, ed è formato da strati e neuroni digitali, che imparano dagli strati inferiori ed estraggono concetti astratti. Si tratta di una gerarchia di complessità crescente. Ogni algoritmo applica una trasformazione non lineare nel suo input ed usa ciò che apprende per creare un modello statistico come output. Ci sono iterazioni fino a quando l'output non ha un livello di precisione accettabile. Nel Deep Learning, il modello impara da solo scoprendo le relazioni tra le variabili. Gli esseri umani non intervengono, il modello è in grado di effettuare la selezione delle caratteristiche. A volte è complicato estrarre caratteristiche astratte di alto livello da dati grezzi. Il Deep Learning risolve questo problema esprimendo queste caratteristiche complesse in termini di combinazioni di quelle più semplici.

Gli sviluppatori si affidano a framework di Deep Learning, come TensorFlow o PyTorch, per attuare la semplificazione del processo di raccolta dati, che vengono usati per il training delle reti neurali di interesse, e per l'implementazione stessa dei modelli di apprendimento più complessi.

Esistono diversi modi per effettuare il training: apprendimento supervisionato e non, per rinforzo. Nello specifico, in questa trattazione, è stato utilizzato l'apprendimento supervisionato; tale processo permette l'esecuzione del training dell'algoritmo sulla base di dati etichettati in precedenza. Utilizzando l'apprendimento supervisionato, l'algoritmo sfrutta le etichette incluse nei dati, forniti come input, per confrontarli con quelli ottenuti dall'algoritmo di analisi e per comprendere se la determinazione effettuata sulle informazioni sia corretta o meno. Con questo tipo di apprendimento è necessario che i dati, su cui viene eseguito il training, vengano forniti dagli esseri umani. La fase di etichettatura dei dati viene effettuata prima che questi vengano usati nella fase di training.

Il Deep Learning viene usato nei settori più vari e per i casi d'uso più disparati; ad esempio, è usato per il riconoscimento di immagini, delle voci e delle emozioni, in servizi di streaming e di e-commerce per favorire raccomandazioni automatizzate e perfezionare le esperienze dei clienti, in chatbot attivati tramite testo o voce, per assistenti digitali personali che devono comprendere il parlato e rispondere in modo appropriato e naturale e, anche, nei veicoli senza conducente per elaborazioni di dati dinamici in tempi rapidi, reagendo più velocemente ad imprevisti rispetto all'uomo.

Una rete neurale è in grado di determinare autonomamente se le previsioni effettuate sono accurate e continua l'apprendimento ed il processo decisionale autonomamente; mentre, un modello di apprendimento automatico richiede l'input umano e basa le proprie decisioni sugli elementi per cui ne è stato eseguito il training.

Si tratta di un sistema che:

- usa livelli non lineari a cascata, l'input di ciascun livello è l'uscita del livello precedente. Si tratta di una rappresentazione gerarchica, in quanto le caratteristiche di più alto livello derivano da quelle di livello inferiore.
- realizza una gerarchia di concetti e apprende multipli livelli di rappresentazione.

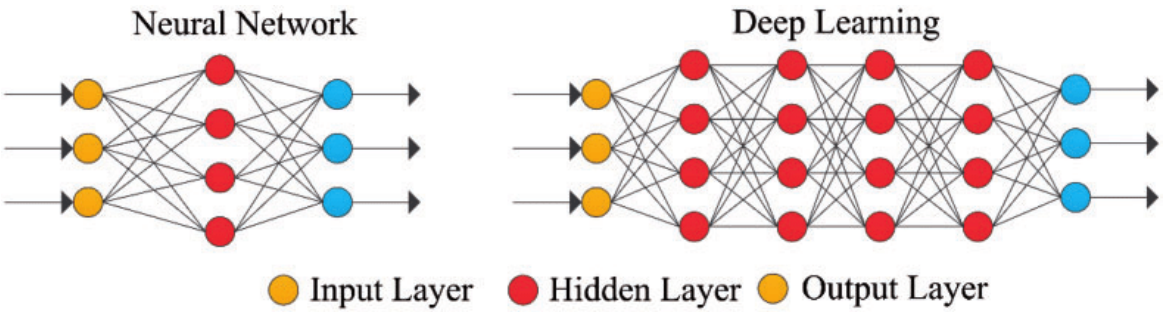


Figura 12 - Neural Network e Deep Learning [2]

Convolutional Neural Networks

Nelle normali reti neurali i neuroni di uno strato sono collegati ai neuroni dello strato precedente ed ogni neurone ha il suo peso specifico; ciò significa che non ci sono ipotesi sui dati immessi nella rete. Questa premessa potrebbe sembrare ottima, se non fosse che, per alcune attività, come quelle legate alle immagini ed al linguaggio, ciò non è efficiente.

Il modello CNN tratta i dati in modo spaziale, ciascun neurone è collegato solo ai neuroni vicini ed hanno tutti lo stesso peso.

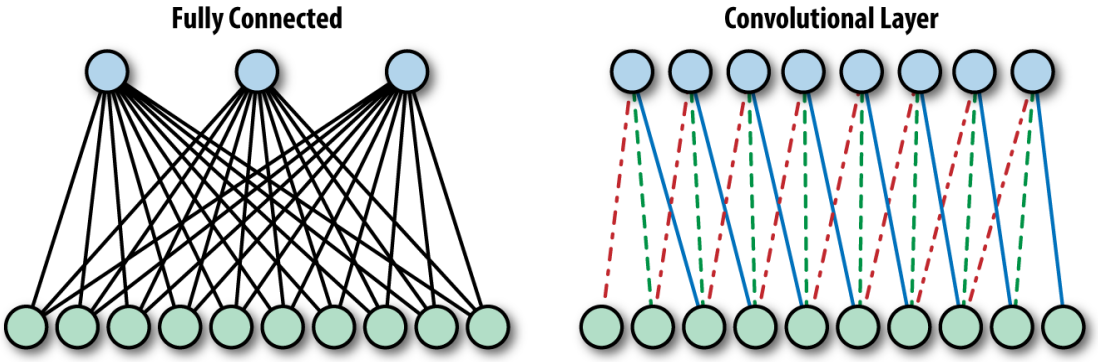


Figura 13 - Differenza tra Fully Connected e Convolutional Layer [3]

Le reti neurali nascono da studi condotti sulle cortecce cerebrali prefrontali degli animali e sono state impiegate nella image recognition già negli anni '80. Similmente a quanto accade nella corteccia visiva, che è una struttura specializzata, dove ogni zona risponde ad una specifica caratteristica del segnale di ingresso, nelle CNNs ogni stadio ha un ruolo specifico.

Un modello CNN è costituito da un blocco di input, un numero variabile di livelli nascosti (hidden layer), che eseguono calcoli tramite funzioni di attivazione ed un livello di output, che effettua la classificazione vera e propria.

Nella rete implementata, proprietà distintiva è la presenza di livelli convoluzionali, che estraggono caratteristiche attraverso l'uso di filtri. I diversi filtri presenti nel livello nascosto rispondono ad una caratteristica specifica del segnale di ingresso. Ogni layer convoluzionale è seguito da un layer di Max-Pooling, che tende ad aumenta il livello di "astrazione". Il pooling aiuta il modello a concentrarsi solo sulle caratteristiche principali di ogni porzione di dati, in più esegue un'aggregazione delle informazioni. Invece, il livello con funzione di attivazione RELU, ovvero Rectified Linear Units, segue i livelli di convoluzione ed ha il compito di annullare i valori non utili, ottenuti nei livelli precedenti. La classificazione è svolta dal livello completamente connesso FC, Fully connected.

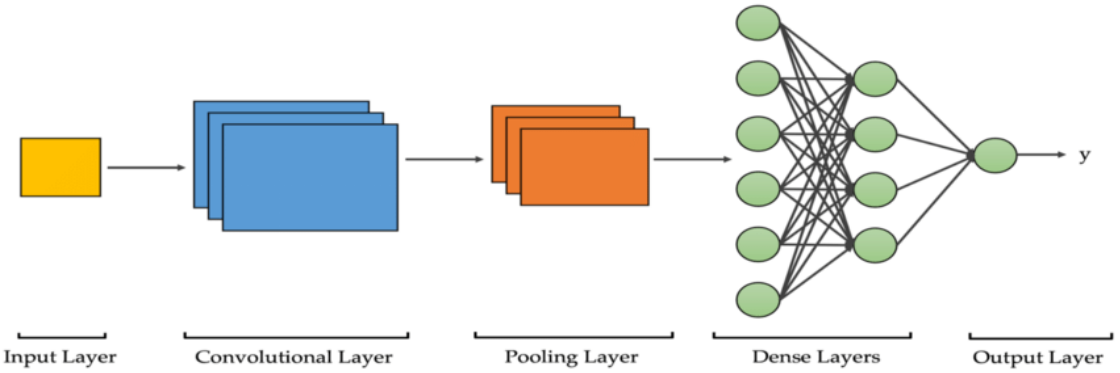


Figura 14 - Convolutional Neural Networks (CNN) [4]

RECURRENT NEURAL NETWORK

Le RNN, reti neurali ricorrenti, nascono negli anni ‘80; le caratteristiche principali sono la presenza di uno stato nascosto ed l’utilizzo di una memoria per ricordare informazioni nel tempo. Il risultato ottenuto nei livelli nascosti viene usato per elaborare l’input futuro. RNN funziona secondo il principio di salvataggio dell'output di un particolare livello ed il reinserimento nell'input per prevedere l'output del livello. Questo tipo di rete nasce con l’obiettivo di risolvere alcuni problemi delle reti neurali feed-forward, come l’impossibilità di memorizzare gli input precedenti, il limite di considerare solo l’input corrente e l’impossibilità di gestire dati sequenziali. Nelle RNN l’informazione scorre attraverso un ciclo fino al livello nascosto.

Ogni neurone può essere interconnesso con uno o più neuroni in modo differente, possono esserci interconnessioni tra neuroni di un livello con quello precedente oppure dei loop. I collegamenti possono essere all’indietro o verso lo stesso livello ed è questa la caratteristica principale, perchè è proprio il concetto di ricorrenza che introduce quello di memoria. I livelli risultano strettamente interconnessi tra loro, infatti i valori di uscita di uno strato superiore sono utilizzati come ingresso da uno strato di livello inferiore. Il comportamento dinamico temporale si presenta quando una sequenza di informazioni, ricevute in istanti precedenti a quello considerato, è usata come input dello strato, che rappresenta la memoria di stato. In altri casi, invece, una dinamica caotica si innesca su strati costituiti da un insieme di neuroni e dotati di loop di connessioni sparse. Tale dinamica è sfruttata per l'addestramento di una parte successiva della rete. Quest’ultimo caso è quello delle echo state networks, usate per il riconoscimento della grafia o il riconoscimento vocale.

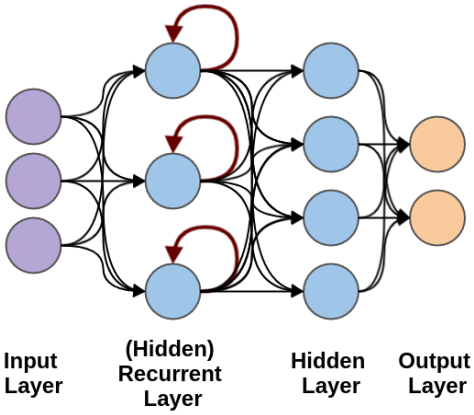


Figura 15 - Esempio generale di Rete Neurale Ricorrente [5]

Come è possibile notare dalla figura precedente, lo strato intermedio ‘h’ può essere costituito da più livelli nascosti; ciascuno con le proprie funzioni di attivazione, pesi e bias.

La rete neurale ricorrente standardizza le diverse funzioni di attivazione, pesi e distorsioni; in modo che ogni livello nascosto abbia gli stessi parametri. Quindi, invece, di creare più livelli nascosti, ne crea uno e lo ripete tutte le volte necessarie.

Esistono diversi modi per l'implementazione di una rete RNN, tra le più rilevanti passiamo ad analizzare quelle che si basano su LSTM.

LSTM Networks

Le reti neurali Long Short Term Memory, note come LSTM, hanno come particolarità quella di poter apprendere e di poter reagire all'evento temporale grazie alla loro connessione di feedback. Questa peculiarità è utile nel caso di applicazioni in cui il tempo è una caratteristica essenziale, come l'elaborazione vocale, la composizione musicale e la descrizione del video. Tuttavia, poiché vengono addestrati utilizzando la propagazione all'indietro nel tempo, i segnali di errore, che scorrono all'indietro nel tempo, possono diventare sempre più grandi o svanire a seconda delle dimensioni dei pesi; ciò creerà pesi oscillanti o renderà la rete lenta nell'addestramento e nella convergenza. Questa nuova architettura delle RNNs è stata introdotta nel 1997 da Hochreiter e Schmidhube per risolvere le problematiche presenti con le RNN. Le RNNs utilizzano le unità ricorrenti, così come anche le LSTM basano il loro funzionamento su queste unità; ma le prime sono molto più semplici delle seconde.

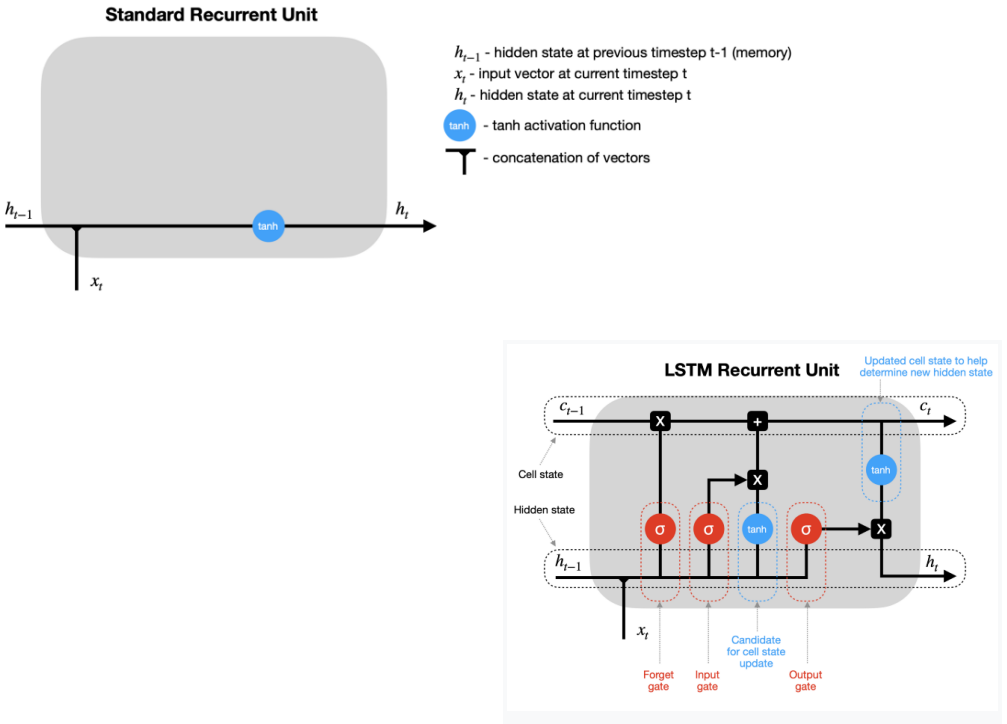


Figura 16 - Standard RNN recurrent unit (in altro a sinistra) e LSTM Recurrent Unit [6]

Come illustrato nella figura 4, nell'unità ricorrente delle RNNs vengono attuate solo due operazioni, la combinazione del nuovo input con lo stato del precedente stato nascosto ed il passaggio del valore ad una funzione di attivazione. Dopo che lo stato nascosto è stato

calcolato al tempo t, questo è passato indietro all'unità ricorrente e combinato con l'input dell'istante successivo; in questo modo lo stato all'istante t+1 può essere calcolato. Questo processo viene ripetuto n volte per tutti gli istanti di tempo catturati. Al contrario, LSTM utilizza vari gates per decidere quali informazioni tenere e quali scartare. Inoltre, è presente una cella di stato, da considerare come la memoria a lungo termine della rete neurale. Le reti LSTM sono in grado di colmare intervalli di tempo maggiori di 1000 passi, ma possono troncare i calcoli del gradiente in un punto definito senza influenzare le attivazioni a lungo termine grazie ad un algoritmo incorporato, che applica un flusso di errore costante attraverso le unità. I sistemi basati su LSTM riescono ad apprendere le caratteristiche spettrali del segnale con buoni risultati.

I componenti fondamentali di una LSTM sono:

- input gate, aiuta a capire quali elementi devono essere aggiunti allo stato della cella.
- output gate, fornisce il risultato.
- forget gate, controlla quali valori nello stato della cella vanno trascurati, moltiplicando il valore per 0, e quali vanno ricordati, moltiplicando per 1 e quali vanno parzialmente ricordati, moltiplicando per un valore tra 0 e 1. La funzione sigmoide, infatti, è compresa tra 0 e 1.
- stato nascosto, lo stato nascosto di uno step e l'input in uno step corrente, vengono combinati prima di far passare copie nei vari gate.
- cella di connessione ricorrente a sé stessa.

Finita la trattazione relativa ai vari tipi di reti neurali esistenti, si passa ad analizzare gli aspetti che riguardano l'addestramento e l'apprendimento di queste.

2.6 Training, validation e testing set

Il pattern è un campione di dati, contenente informazioni utili. Per condurre l'addestramento e la valutazione di un modello ottimale si utilizzano e si distinguono tre diversi set:

- training set, rappresenta il set di dati iniziale usato per addestrare gli algoritmi di apprendimento. I modelli creano e raffinano le proprie regole su questi dati.
- validation set, campione di dati usato per la valutazione imparziale di un adattamento del modello al set di dati di addestramento durante l'ottimizzazione degli iperparametri. La valutazione diventa più parziale man mano che l'abilità sul set di dati di convalida viene incorporata nella configurazione del modello.
- testing set, campione di dati usato per la valutazione imparziale di un adattamento del modello e delle prestazioni finali del sistema.

Per evitare che le prestazioni del modello vengano sopravvalutate e per limitare situazioni di overfitting vengono scelti campioni di dati disgiunti. Se i pattern non risultano disgiunti, potrebbero verificarsi i seguenti casi:

- Training set contenente validation set e testing set: gli esempi forniti in fase di valutazione sono già stati visti ed "imparati" dal modello.
- Validation set contenente dati mai visti dal modello: buona risposta del modello, che riesce a raggiungere un buon livello di generalizzazione durante la fase di addestramento.

2.7 Model fit

La ricerca della funzione obiettivo ottimale relativa al modello fittato è l'obiettivo che si vuole raggiungere durante la fase di addestramento. Un'epoca significa che ogni campione nel set di dati di addestramento ha avuto l'opportunità di aggiornare i parametri del modello interno. Un'epoca è composta da uno o più batches. Il numero di epoche è un iperparametro, che definisce il numero di volte in cui l'algoritmo di apprendimento funzionerà attraverso l'intero set di dati di addestramento. Bisogna eseguire più cicli di elaborazione per riuscire ad elaborare l'intero training set, un'epoca termina quando l'intero set è stato sottoposto al modello. La fase di training, solitamente, si esaurisce dopo diverse epoches (possono servirne anche decine di migliaia). È importante, comunque, tenere a mente che aumentare il numero di epoche, non significa sempre migliorare il modello.

Con il termine batch è indicato un iperparametro, che definisce il numero di samples con cui lavorare in fase di training. Il training set viene diviso in uno o più batches. Un batch è da pensare come ad un ciclo for, che esegue un'iterazione su uno o più campioni e fa previsioni.

Alla fine del batch, le previsioni vengono confrontate con le variabili di output previste e viene calcolato un errore. Quando il batch ha le dimensioni di un campione, l'algoritmo di apprendimento è chiamato discesa del gradiente stocastico. Quando la dimensione del batch è più di un campione, ma inferiore alla dimensione del set di dati di addestramento, l'algoritmo di apprendimento viene chiamato discesa del gradiente mini-batch. Quando il set di training è molto grande e, quindi, la sua elaborazione non può essere condotta in un singolo step, risulta utile tale suddivisione. Il numero di esempi contenuti in ogni batch è chiamato batch size. Per completare un'epoca è necessario un numero variabile di batches.

Se la suddivisione in batch non è fatta uniformemente oppure se l'assegnazione dei valori non viene fatta correttamente, le problematiche verso cui si può incorrere sono:

- non raggiungimento delle buone performance, se il valore del batch size è troppo piccolo, in quanto in questo caso sarà piccola la quantità di dati da elaborare.
- overfitting o esaurimento della memoria, se il valore di batch size è troppo grande.

La velocità del modello ed il suo modo di trovare una regola generale a partire dai dati che gli vengono forniti dipendono, anche, dal numero di epoche e dal valore di batch definito.

2.8 Metriche

Le metriche, strumenti matematici e/o statistici, entrano in gioco quando si deve attuare una quantificazione della differenza tra il risultato ottenuto dal sistema e quello atteso. Nello specifico, tali strumenti permettono una valutazione di quanto la risposta del sistema analizzato sia lontana da quella corretta. Dalla scelta della giusta metrica deriva l'attendibilità del risultato ed anche la velocità con cui l'addestramento di una rete viene portato avanti.

2.8.1 Loss e Accuracy

La loss è una metrica fondamentale per capire se ci si trova di fronte ad una buona o cattiva previsione; nello specifico, la loss indica la perdita e, quindi, quanto cattiva è la previsione su un esempio. È la differenza tra il valore effettivo ed il valore previsto. Se la previsione del modello è perfetta, la perdita è zero; in caso contrario, la perdita è maggiore. L'obiettivo dell'addestramento di un modello è trovare un insieme di pesi e distorsioni che abbiano una bassa perdita, in media, in tutti gli esempi. La loss viene ricavata dai set di validation e training ed il suo valore è una somma degli errori commessi.

L'accuracy misura le prestazioni di un modello e viene ricavata dopo che, a seguito del fit del modello, i diversi parametri sono stati appresi. In un sistema di classificazione l'accuratezza è da definire come il rapporto tra i campioni correttamente elaborati ed il totale dei campioni inviati al modello. L'accuracy è espressa in percentuale.

2.8.2 Matrice di confusione

Un altro modo per valutare le prestazioni della classificazione, realizzata da un modello, è l'utilizzo della Confusion Matrix, matrice di confusione. È una matrice N x N, dove N è il numero di classi target; la variabile target può assumere due differenti valori: Positivo o Negativo. La matrice confronta i valori target effettivi con quelli previsti dal modello. Questa rappresentazione ci permette di identificare il tipo di errori che sta commettendo il modello. In questa mappa bidimensionale, le colonne rappresentano i valori effettivi della variabile target, mentre le righe rappresentano i valori previsti della variabile target. La colorazione delle celle è conseguenza del rapporto tra classe predetta e classe effettiva; nello specifico, vengono evidenziate con colori scuri le zone di confusione, dove il modello non si comporta adeguatamente, e con sfumature di diverso colore zone, dove il modello predice più o meno bene la classe. La matrice di confusione diventa un modo diretto per visualizzare come siano distribuiti gli errori e, quindi, se il problema sia del modello o del training set poco equilibrato.

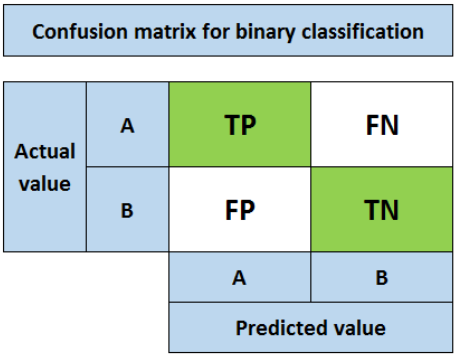


Figura 17 - Esempio Confusion Matrix [7]

2.8.3 Precision e Recall

Per quanto riguarda la classificazione di un pattern i risultati ottenibili sono:

- True Positive (vero positivo) quando il modello prevede correttamente la classe positiva, un pattern è positivo e assegnato al valore positivo.

- True Negative (vero negativo) quando il modello prevede correttamente la classe negativa, un pattern negativo è riconosciuto come tale.
- False Positive (falso positivo) quando ad un pattern negativo il modello assegna erroneamente la classe positiva.
- False Negative (falso negativo) quando ad un pattern positivo è assegnata erroneamente la classe negativa dal modello.

La precisione è calcolata come rapporto tra il numero di campioni Positivi correttamente classificati e il numero totale di campioni classificati come Positivi (correttamente o erroneamente). La precisione misura l'accuratezza del modello nel classificare un campione come positivo.
La recall è calcolata come rapporto tra il numero di campioni positivi correttamente classificati come positivi ed il numero totale di campioni positivi. Questo parametro misura la capacità del modello di rilevare campioni positivi. Maggiore è la recall, ovvero il richiamo, maggiore è il numero di campioni rilevati come positivi.
Infine, F1 è una metrica che cattura sia la tendenza della recall sia quella della precision, è una media armonica di entrambe ed è massimo il suo valore quando la recall è uguale alla precision.

$$\mathrm{PR} = \frac{\mathrm{TP}}{\mathrm{TP+FP}}$$
$$\mathrm{RE} = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$$
$$\mathrm{CA} = \frac{\mathrm{TP+TN}}{\mathrm{TP+TN+FP+FN}}$$
$$\mathrm{F_1} = \frac{2\mathrm{TP}}{2\mathrm{TP+FP+FN}}$$

In generale, in fase di addestramento la loss dovrebbe registrare un andamento decrescente, con tendenza asintotica ad un valore minimo; mentre l'accuracy dovrebbe mostrare un andamento crescente, con tendenza asintotica a 1. Il comportamento della loss e dell'accuracy permettono di capire le caratteristiche dell'addestramento del modello in esame. In molti casi la convergenza non si verifica, ad esempio, se sono presenti troppe oscillazioni tra valori alti e bassi della loss, può essere che l'algoritmo di ottimizzazione dei parametri non sia giusto oppure che gli iperparametri non siano ottimali. Altro caso si ha quando l'accuracy resta lontana da 1, è possibile che il modello non sia in grado di gestire la complessità. Ultimo caso si ha quando la loss decresce e l'accuracy non cresce, in questa situazione può essere che le metriche non siano corrette o che il modello stia overfittando.

2.8.4 Overfitting e Underfitting

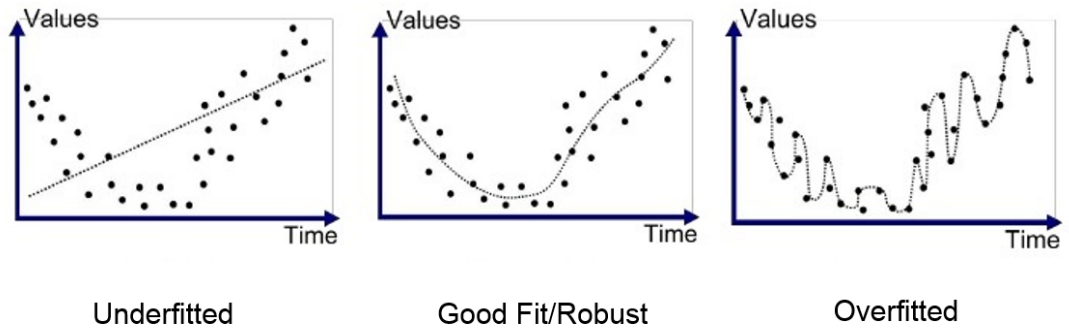


Figura 18 - Overfitting, Underfitting [8]

La condizione di overfitting si verifica quando l'accuracy calcolata sul training set tende ad 1, mentre l'accuracy sul validation set o sul testing set tende ad allontanarsi da 1. In una situazione di overfitting il modello, piuttosto, che imparare una regola generale per il riconoscimento, impara a memoria i pattern forniti in input. Il modello riconosce perfettamente i pattern già visti, ma va facilmente in errore quando viene posto di fronte a pattern mai visionati. Quando il modello si allena per un intervallo di tempo troppo lungo sui dati di esempio o quando risulta troppo complesso o quando il training set è troppo piccolo e/o poco vario, si verifica una situazione di apprendimento di "rumore" all'interno del set di dati. Questa situazione si traduce come apprendimento di dati non rilevanti, quando il modello memorizza tale rumore e si adatta troppo al set di addestramento, diventa "overfittato" e perde la sua capacità nel trovare una regola generale. In caso di overfitting si può intervenire modificando la complessità del modello o applicando tecniche di dropout. L'operazione più semplice, comunque, rimane l'interruzione dell'addestramento, l'epoch in cui si interrompe l'addestramento è chiamata early stopping epoch e rappresenta il punto in cui si raggiunge il miglior risultato con la configurazione testata.

L'underfitting è un problema di apprendimento, che si verifica quando sono pochi i parametri su cui si basa la classificazione e c'è una grande discrepanza (high bias). Il problema non dipende dalla casualità dei dati di training, ma dall'eccessiva semplicità del modello di classificazione e dalla semplicità dell'apprendimento. Per risolvere l'underfitting si deve agire sul modello per aumentarne la complessità.

Sia nel caso di overfitting che in quello di underfitting sono raggiunte buone conclusioni sui dati di addestramento, ma si ottengono scarse performance sui dati di test.

Solitamente man mano che aumentano le epoche, la loss diminuisce e l'accuracy aumenta, ma con val_loss e val_acc possono verificarsi diversi casi:

- val_loss aumenta e val_accuracy diminuisce, questo significa che il modello si sta riempiendo di valori senza imparare.
- val_loss aumenta e val_acc aumenta, questo potrebbe significare un caso di overfitting.
- val_loss diminuisce e val_acc aumenta, questo è il caso ottimale in quanto significa che il modello sta imparando e sta funzionando correttamente.

Parte sperimentale

Il riconoscimento vocale delle emozioni ha occupato negli anni un ruolo di grande rilievo nelle ricerche dei data scientist. Sono molti i repository consultabili in rete ed è a partire da questi, che sono stati condotti i vari test contenuti nella tesi. Inoltre, sono stati estrapolati risultati, che permettono valutazioni approfondite sugli algoritmi utilizzati, sulle metriche che maggiormente influenzano i loro valori e le variazioni dei modelli con i cambiamenti più significativi ai fini degli obiettivi prefissati.

Tutti i test condotti sono consultabili al link https://www.dropbox.com/s/1h5ct6rz5xhlmin/TEST.zip?dl=0

3.1 Test su CNN

A partire dal repository https://github.com/MITESHPUTHRANNEU/Speech-Emotion-Analyzer, repository con maggior numero di valutazioni positive su Github riguardante lo Speech Emotion Recognition, sono stati condotti dei test su una rete neurale CNN. Il modello analizzato è del tipo:

```
model = Sequential()

model.add(Conv1D(256, 5,padding='same',
                 input_shape=(x_traincnn.shape[1],x_traincnn.shape[2])))
model.add(Activation('relu'))
model.add(Conv1D(128, 5,padding='same'))
model.add(Activation('relu'))
model.add(Dropout(0.1))
model.add(MaxPooling1D(pool_size=(8)))
model.add(Conv1D(128, 5,padding='same',))
model.add(Activation('relu'))
model.add(Conv1D(128, 5,padding='same',))
model.add(Activation('relu'))
model.add(Flatten())
model.add(Dense(y_train.shape[1]))
model.add(Activation('softmax'))
opt = keras.optimizers.RMSprop(lr=0.00001, decay=1e-6)
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 65, 256)           1536

 activation (Activation)     (None, 65, 256)           0

 conv1d_1 (Conv1D)           (None, 65, 128)           163968

 activation_1 (Activation)   (None, 65, 128)           0

 dropout (Dropout)           (None, 65, 128)           0

 max_pooling1d (MaxPooling1D  (None, 8, 128)           0
 )

 conv1d_2 (Conv1D)           (None, 8, 128)            82048

 activation_2 (Activation)   (None, 8, 128)            0

 conv1d_3 (Conv1D)           (None, 8, 128)            82048

 activation_3 (Activation)   (None, 8, 128)            0

 flatten (Flatten)           (None, 1024)              0

 dense (Dense)               (None, 7)                 7175

 activation_4 (Activation)   (None, 7)                 0

=================================================================
Total params: 336,775
Trainable params: 336,775
Non-trainable params: 0
```

Figura 19 - CNN model

Nella rete studiata il primo layer è quello definito come Conv1D, livello in grado di creare un kernel di convoluzione, che è convogliato con l'input del livello su una singola dimensione spaziale (o temporale) per produrre un tensore di output. Conv1D è usato quando si fa scorrere il kernel lungo un'unica dimensione, mentre Conv2D quando gli stessi pesi vengono fatti scorrere lungo due dimensioni, come nel caso di un'immagine 2D. Nel nostro caso i livelli di convoluzione inseriti nella rete sono tutti di tipo Conv1D, in quanto quella condotta è una convoluzione nel tempo, trattandosi di audio e features, che hanno il loro sviluppo nel tempo. Oltre al primo livello di convoluzione, altri livelli di convoluzione vengono inseriti nel modello, nel caso specifico i layers convoluzionali in totale risultano essere 4. Notiamo che nei livelli convoluzionali il valore padding è settato a 'same'. Il padding può essere di due diversi tipi, esso può assumere valore 'valid' oppure 'same'. Nel caso specifico 'same' significa che la dimensione dell'output è la stessa della dimensione dell'input. L'uso di 'same' assicura che il filtro venga applicato a tutti gli elementi dell'input. Al contrario, il valore di padding "valid" indica che non c'è alcun padding, ovvero non si verifica alcun riempimento. La dimensione dell'output dello strato convoluzionale si riduce a seconda della dimensione dell'input e della dimensione del kernel. La finestra del filtro rimane in una posizione valida all'interno della mappa di input, quindi la dimensione dell'output si riduce di filter_size-1. Dopo la convoluzione, c'è il pooling che permette di ridurre le dimensioni, estraendo solo le features più utili. Il parametro pool_size è un intero e setta la dimensione della finestra di MaxPooling. Non è possibile passare l'output dello strato convoluzionale direttamente allo strato denso, perché l'output dello strato convoluzionale è in forma multidimensionale e lo strato denso richiede input in forma unidimensionale; per questo motivo si usa il metodo Flatten() tra i due livelli. Il metodo Flatten() converte la matrice multidimensionale in matrice unidimensionale.

Lo strato denso è un semplice strato di neuroni in cui ogni neurone riceve input da tutti i neuroni dello strato precedente. Il risultato del flatten viene inviato al livello fully connected, che ne effettua la classificazione. Ogni strato nella rete neurale contiene neuroni, che calcolano la media pesata del suo input e questa media pesata viene passata attraverso la "funzione di attivazione". Il risultato di questa funzione non lineare viene trattato come output di quel neurone. La funzione Relu è la miglior funzione nel campo del Deep Learning. Da vari esperimenti si è dedotto che l'addestramento di una rete profonda con ReLu tende a convergere molto più rapidamente ed in modo più affidabile rispetto all'addestramento di una rete profonda con attivazione sigmoidea.

Per quanto riguarda l'estrazione delle features, questa viene condotta tramite la libreria 'librosa' in Python, che è una delle librerie più utilizzate nell'analisi audio. Dataset di partenza è Berlin e lo splitting è realizzato mantenendo sempre la divisione dell' 80% per il train set e del 20% per il test set.

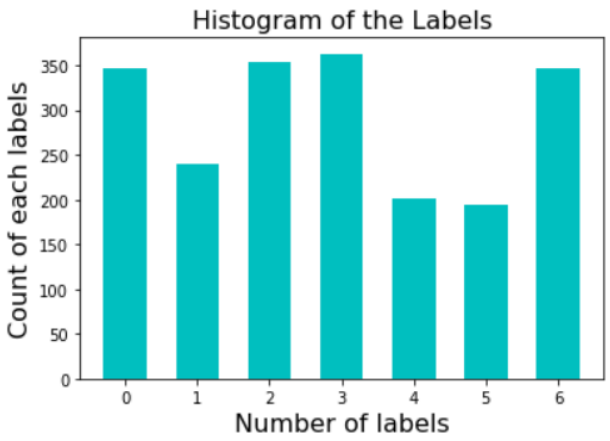
```
dataset_path = os.path.abspath('./Dataset')
destination_path = os.path.abspath('./')
# To shuffle the dataset instances/records
randomize = True
# for spliting dataset into training and testing dataset (80/20%)
split = 0.8
# Number of sample per second e.g. 16KHz
sampling_rate = 20000
emotions=["anger","disgust","fear","happy","neutral", "sad", "surprise"]
```

```
# loading dataframes using dataset module
from utils import dataset
df, train_df, test_df = dataset.create_and_load_meta_csv_df(dataset_path, destination_path, randomize, split)
```

Dalle operazioni sopra mostrate si ottiene:

```
Dataset samples  :  2556
Training Samples :  2044
testing Samples  :  512
```

Le emozioni sono distinte in 7 classi e il numero delle etichette, assegnate ad ogni audio per ciascuna emozione, è rappresentato dall'istogramma di seguito.



Le features estratte da ogni rappresentazione audio sono mostrate nella figura 20.

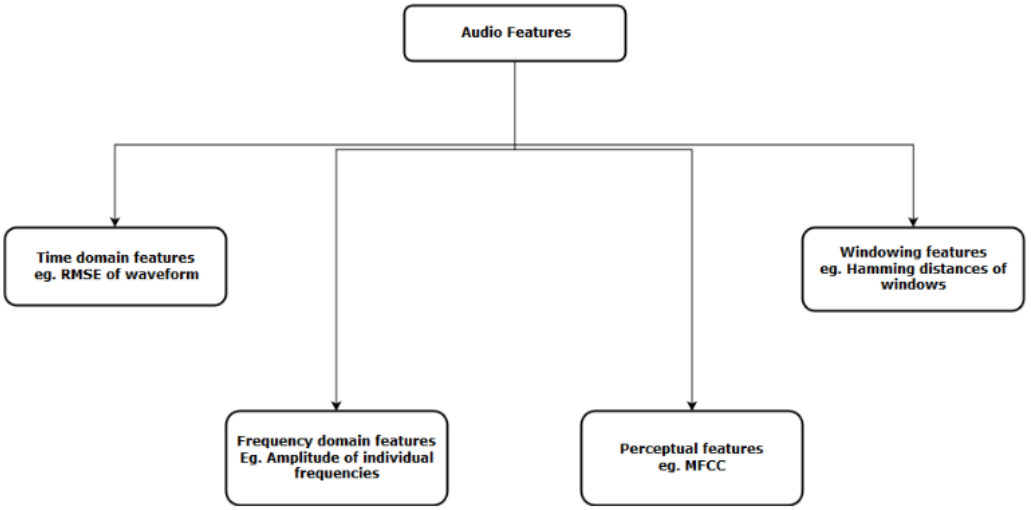


Figura 20 - Features dell'audio [9]

A seguito dell'estrazione delle features, queste sono inviate al modello per le analisi, si passa al fit ed all'evaluate del modello 1D CNN.

In particolare, l'estrazione delle features viene effettuata tramite le funzioni seguenti:

```python
# feature_extracting
import librosa
import pandas as pd
import numpy as np

def get_audio_features(audio_path,sampling_rate):
    X, sample_rate = librosa.load(audio_path ,res_type='kaiser_fast',duration=2.5,sr=sampling_rate*2,offset=0.5)
    sample_rate = np.array(sample_rate)

    y_harmonic, y_percussive = librosa.effects.hpss(X)
    pitches, magnitudes = librosa.core.pitch.piptrack(y=X, sr=sample_rate)

    mfccs = np.mean(librosa.feature.mfcc(y=X,sr=sample_rate,n_mfcc=13),axis=1)

    pitches = np.trim_zeros(np.mean(pitches,axis=1))[:20]

    magnitudes = np.trim_zeros(np.mean(magnitudes,axis=1))[:20]

    C = np.mean(librosa.feature.chroma_cqt(y=y_harmonic, sr=sampling_rate),axis=1)

    return [mfccs, pitches, magnitudes, C]


def get_features_dataframe(dataframe, sampling_rate):
    labels = pd.DataFrame(dataframe['label'])

    features  = pd.DataFrame(columns=['mfcc','pitches','magnitudes','C'])
    for index, audio_path in enumerate(dataframe['path']):
        features.loc[index] = get_audio_features(audio_path, sampling_rate)

    mfcc = features.mfcc.apply(pd.Series)
    pit = features.pitches.apply(pd.Series)
    mag = features.magnitudes.apply(pd.Series)
    C = features.C.apply(pd.Series)

    combined_features = pd.concat([mfcc,pit,mag,C],axis=1,ignore_index=True)

    return combined_features, labels
```

Il calcolo dei coefficienti MFCCs viene portato avanti tenendo sempre in considerazione il parametro di duration e quello di n_mfcc. Nell'estrazione delle features viene specificata la duration, in modo che venga considerato l'audio per una quantità di tempo definita, in questo caso 2.5 secondi. Invece, l'offset indica dopo quanto tempo bisogna iniziare la lettura. Il parametro sr (sampling rate) indica la frequenza di campionamento, ovvero il numero di campioni al secondo prelevati da un segnale; mentre il parametro n_mfcc indica il numero di MFCCs da ritornare.

Ora che è stato illustrato approfonditamente il punto di partenza fornito dal repository, verranno mostrate le varie modifiche fatte sul codice di partenza in modo da interpretare le conclusioni derivate dalle diverse scelte fatte.

3.1.1 Risultati ottenuti su CNN

Nel primo test svolto con la rete presentata nel paragrafo precedente, si è utilizzato come dataset di partenza Berlin, database di dimensione modesta, con emozioni etichettate nel seguente modo:

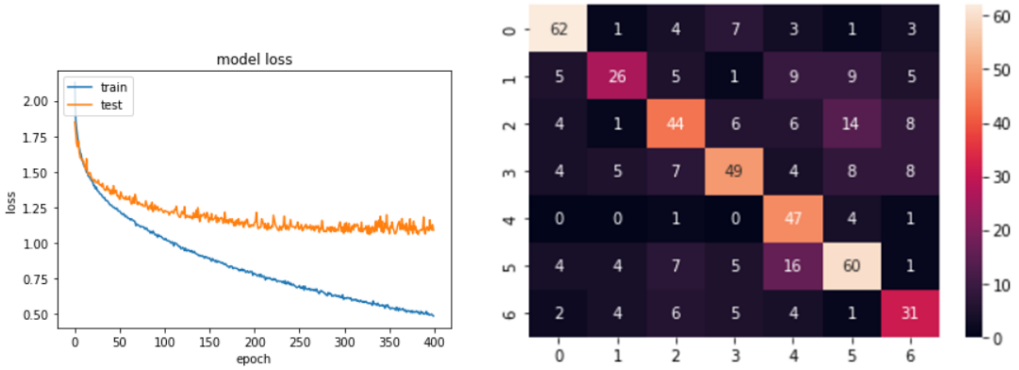
- 0 : anger
- 1 : disgust
- 2 : fear
- 3 : happy
- 4 : neutral
- 5 : sad
- 6 : surprise

A seguito del fit del modello

```
cnnhistory=model.fit(x_traincnn, y_train, batch_size=16, epochs=400, validation_data=(x_testcnn, y_test))
```

vengono raggiunti i seguenti valori:

```
Epoch 398/400
128/128 [==============================] - 3s 26ms/step - loss: 0.4948 - accuracy: 0.8337 - val_loss: 1.0890 - val_accuracy:
0.5918
Epoch 399/400
128/128 [==============================] - 3s 26ms/step - loss: 0.4850 - accuracy: 0.8356 - val_loss: 1.1246 - val_accuracy:
0.5938
Epoch 400/400
128/128 [==============================] - 3s 27ms/step - loss: 0.4818 - accuracy: 0.8386 - val_loss: 1.0863 - val_accuracy:
0.6230
```



accuracy del modello: 62.30%

	precision	recall	f1-score	support
0	0.77	0.77	0.77	81
1	0.63	0.43	0.51	60
2	0.59	0.53	0.56	83
3	0.67	0.58	0.62	85
4	0.53	0.89	0.66	53
5	0.62	0.62	0.62	97
6	0.54	0.58	0.56	53
accuracy			0.62	512
macro avg	0.62	0.63	0.62	512
weighted avg	0.63	0.62	0.62	512

0	0.765432
1	0.433333
2	0.530120
3	0.576471
4	0.886792
5	0.618557
6	0.584906

Dal test svolto è possibile concludere dicendo che i risultati ottenuti sono buoni, in quanto il valore di accuracy generale raggiunto è del 62.30%. Nello specifico, dalla confusion matrix notiamo che le predizioni sono buone, a meno di alcune classi. Il modello di apprendimento confonde maggiormente: la classe 1 con la 4 e la 5, la classe 2 con la 5 e la 6, la classe 3 con la 5 e la 6, la classe 5 è confusa con la 4. Pertanto, il disgusto è confuso per tristezza oppure emozione neutra; la paura è confusa per tristezza o sorpresa; la gioia è confusa per tristezza e sorpresa; ed, infine, la tristezza è confusa per emozione neutra. Inoltre, il modello fa fatica anche a distinguere la rabbia con la gioia. Nonostante queste interpretazioni errate da parte del modello, se ci si focalizza sull'accuracy per classe è facile notare che le criticità da considerare maggiormente sono quelle relative al disgusto, alla paura e alla tristezza. I risultati ottenuti dai test sono, comunque, coerenti con quanto spiegato nel background teorico del capitolo precedente. È coerente che le interpretazioni della gioia e della paura risultino difficili, in quanto entrambe sono identificate da un tono acuto. Allo stesso modo paura e tristezza vengono confuse per il fatto che la voce risulta stretta ed il ritmo lento; ritmo lento che contraddistingue anche l'emozione neutra e comporta una difficoltà nella classificazione tra emozione neutra e tristezza. Infine, la gioia in alcuni casi, è confusa per sorpresa, dato che entrambe hanno intensità elevate e voce ampia e velocità alta; in altri casi è confusa per collera visto che entrambe presentano tono ed intensità elevate e velocità ancora una volta alta.

Anche il ruolo rappresentativo del classification report è di grande rilievo ai fini della valutazione del riconoscimento vocale delle emozioni.

Nel classification report il valore che ha maggiore rilievo è F1-score, che misura l'accuratezza di un test sulla base della precisione e del richiamo. Ricordiamo che, la precisione è il numero di risultati veri positivi diviso il numero di tutti i risultati positivi, compresi quello non identificati correttamente; mentre il richiamo è il numero di risultati veri positivi diviso per il numero di tutti i campioni, che avrebbero dovuto essere identificati come positivi. Detto ciò, il valore F1 è la media armonica della precisione e del richiamo. Notiamo che i valori di F1 sono buoni per tutte le classi, in particolar modo il valore più elevato di F1 si registra per la classe 0, ovvero per la rabbia; mentre, il valore più basso si ha per la classe 1, ovvero il disgusto.

I risultati raggiunti con questo modello sono buoni, ma sono state apportate modifiche di vario tipo, per vedere se fosse possibile raggiungere risultati migliori e per dedurre quali parametri influenzano maggiormente il risultato finale di predizione. Prima modifica attuata è stata quella dello splitting del dataset.

- ***Modifica splitting dati N=2 e introduzione EarlyStopping***

  Oltre allo split del dataset in training set e testing set con rapporto 80:20, tramite la metodologia StratifiedKFold è stato creato un terzo set di dati, ovvero il validation set. Questa tecnica, denominata anche cross validation, è applicata con l'obiettivo di eliminare possibili problemi di overfitting nel training set. Dalla procedura attuata, derivano N alberi decisionali parziali con un determinato valore predittivo.

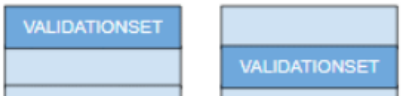


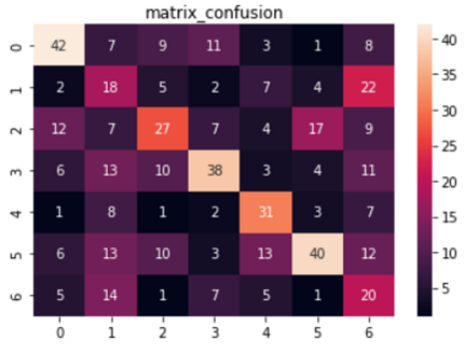
Figura 21 - Esempio di splitting con N=2

A seguito di questa operazione, segue il calcolo della media dei valori predittivi dei K esperimenti e si seleziona l'albero parziale con il valore predittivo più alto.

I dati trovati con questa prima modifica sono mostrati di seguito:

```
Epoch 72/700
64/64 [==============================] - 3s 43ms/step - loss: 1.2144 - accuracy: 0.5744 - val_loss: 1.3795 - val_accuracy: 0.
4618
Evaluate on test data
32/32 [==============================] - 1s 11ms/step - loss: 1.2182 - accuracy: 0.5793
```

accuracy: 42.18%



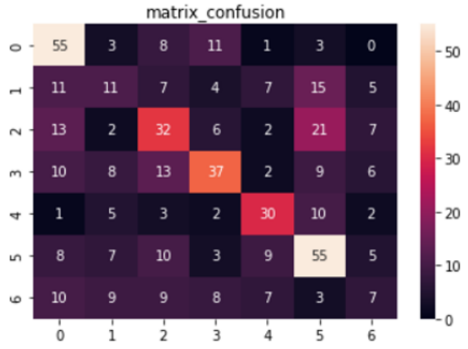
classification

	precision	recall	f1-score	support
0	0.57	0.52	0.54	81
1	0.23	0.30	0.26	60
2	0.43	0.33	0.37	83
3	0.54	0.45	0.49	85
4	0.47	0.58	0.52	53
5	0.57	0.41	0.48	97
6	0.22	0.38	0.28	53
accuracy			0.42	512
macro avg	0.43	0.42	0.42	512
weighted avg	0.46	0.42	0.43	512

0	0.518519
1	0.366667
2	0.325301
3	0.447059
4	0.584906
5	0.412371
6	0.377358

```
Epoch 11/700
64/64 [==============================] - 2s 35ms/step - loss: 1.3381 - accuracy: 0.4814 - val_loss: 1.2341 - val_accuracy: 0.
5616
Evaluate on test data
32/32 [==============================] - 1s 10ms/step - loss: 1.3657 - accuracy: 0.4726
```

accuracy: 44.34%



	precision	recall	f1-score	support
0	0.51	0.68	0.58	81
1	0.24	0.18	0.21	60
2	0.39	0.39	0.39	83
3	0.52	0.44	0.47	85
4	0.52	0.57	0.54	53
5	0.47	0.57	0.52	97
6	0.22	0.13	0.16	53
accuracy			0.44	512
macro avg	0.41	0.42	0.41	512
weighted avg	0.43	0.44	0.43	512

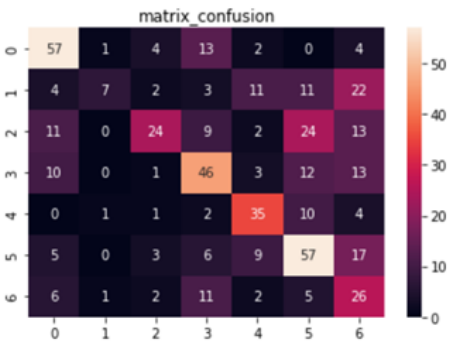
0	0.679012
1	0.250000
2	0.385542
3	0.435294
4	0.566038
5	0.567010
6	0.188679

Si osserva che i valori di accuracy, sia quelli generali che quelli delle singole classi, sono diminuiti. Si continuano ad avere buoni risultati per alcune classi, come le classi 0,3,4,5; mentre le classi che già presentavano problematiche nell'essere riconosciute, adesso presentano valori di accuracy ancora più bassi, per l'esattezza in un range tra 18% e 38%.

- ***Modifica splitting dati N=7 e introduzione EarlyStopping***

```
Epoch 80/700
110/110 [==============================] - 2s 15ms/step - loss: 1.0781 - accuracy: 0.5965 - val_loss: 1.2941 - val_accuracy:
0.5068
Evaluate on test data
55/55 [==============================] - 0s 6ms/step - loss: 1.0763 - accuracy: 0.6005
```

accuracy: 49.21%



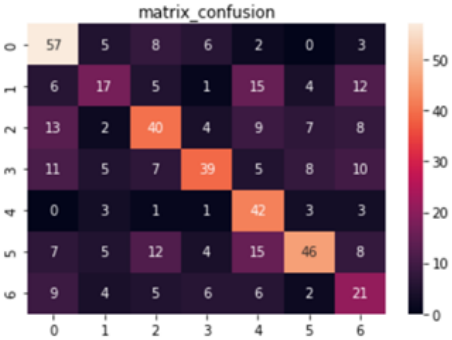
classification

	precision	recall	f1-score	support
0	0.61	0.70	0.66	81
1	0.70	0.12	0.20	60
2	0.65	0.29	0.40	83
3	0.51	0.54	0.53	85
4	0.55	0.66	0.60	53
5	0.48	0.59	0.53	97
6	0.26	0.49	0.34	53
accuracy			0.49	512
macro avg	0.54	0.48	0.46	512
weighted avg	0.54	0.49	0.48	512

0	0.580247
1	0.300000
2	0.421687
3	0.447059
4	0.716981
5	0.556701
6	0.433962

```
Epoch 12/700
110/110 [==============================] - 2s 15ms/step - loss: 1.1095 - accuracy: 0.6005 - val_loss: 1.0236 - val_accuracy:
0.6233
Evaluate on test data
55/55 [==============================] - 0s 6ms/step - loss: 1.1075 - accuracy: 0.6090
```

accuracy: 51.18%



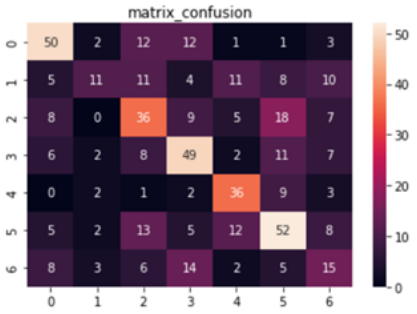
classification

	precision	recall	f1-score	support
0	0.55	0.70	0.62	81
1	0.41	0.28	0.34	60
2	0.51	0.48	0.50	83
3	0.64	0.46	0.53	85
4	0.45	0.79	0.57	53
5	0.66	0.47	0.55	97
6	0.32	0.40	0.36	53
accuracy			0.51	512
macro avg	0.51	0.51	0.50	512
weighted avg	0.53	0.51	0.51	512

0	0.530864
1	0.416667
2	0.481928
3	0.435294
4	0.660377
5	0.474227
6	0.396226

```
Epoch 13/700
110/110 [==============================] - 2s 15ms/step - loss: 1.0795 - accuracy: 0.6096 - val_loss: 1.1621 - val_accuracy:
0.5685
Evaluate on test data
55/55 [==============================] - 0s 7ms/step - loss: 1.0575 - accuracy: 0.6404
```

accuracy:48.63%



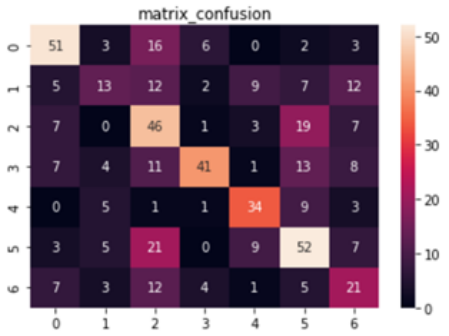
classification

	precision	recall	f1-score	support
0	0.61	0.62	0.61	81
1	0.50	0.18	0.27	60
2	0.41	0.43	0.42	83
3	0.52	0.58	0.54	85
4	0.52	0.68	0.59	53
5	0.50	0.54	0.52	97
6	0.28	0.28	0.28	53
accuracy			0.49	512
macro avg	0.48	0.47	0.46	512
weighted avg	0.49	0.49	0.48	512

0	0.691358
1	0.250000
2	0.578313
3	0.505882
4	0.660377
5	0.350515
6	0.226415

```
Epoch 29/700
110/110 [==============================] - 2s 14ms/step - loss: 1.0082 - accuracy: 0.6324 - val_loss: 1.0528 - val_accuracy:
0.6301
Evaluate on test data
55/55 [==============================] - 0s 6ms/step - loss: 0.9980 - accuracy: 0.6684
```

accuracy: 50.39%



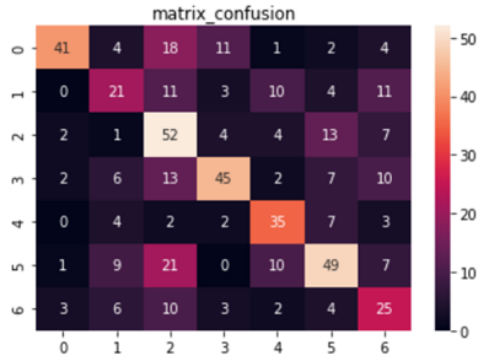
classification

	precision	recall	f1-score	support
0	0.64	0.63	0.63	81
1	0.39	0.22	0.28	60
2	0.39	0.55	0.46	83
3	0.75	0.48	0.59	85
4	0.60	0.64	0.62	53
5	0.49	0.54	0.51	97
6	0.34	0.40	0.37	53
accuracy			0.50	512
macro avg	0.51	0.49	0.49	512
weighted avg	0.52	0.50	0.50	512

0	0.679012
1	0.183333
2	0.506024
3	0.529412
4	0.584906
5	0.463918
6	0.245283

```
Epoch 16/700
110/110 [==============================] - 2s 14ms/step - loss: 0.9858 - accuracy: 0.6455 - val_loss: 1.0771 - val_accuracy:
0.5719
Evaluate on test data
55/55 [==============================] - 0s 6ms/step - loss: 0.9881 - accuracy: 0.6650
```

accuracy: 52.34%



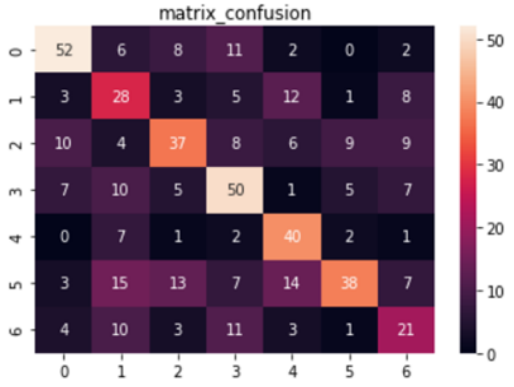
classification

	precision	recall	f1-score	support
0	0.84	0.51	0.63	81
1	0.41	0.35	0.38	60
2	0.41	0.63	0.50	83
3	0.66	0.53	0.59	85
4	0.55	0.66	0.60	53
5	0.57	0.51	0.54	97
6	0.37	0.47	0.42	53
accuracy			0.52	512
macro avg	0.54	0.52	0.52	512
weighted avg	0.56	0.52	0.53	512

0	0.580247
1	0.416667
2	0.385542
3	0.482353
4	0.660377
5	0.515464
6	0.415094

```
Epoch 19/700
110/110 [==============================] - 2s 15ms/step - loss: 0.9662 - accuracy: 0.6450 - val_loss: 0.9423 - val_accuracy:
0.6404
Evaluate on test data
55/55 [==============================] - 0s 6ms/step - loss: 0.9636 - accuracy: 0.6741
```

accuracy: 51.95%



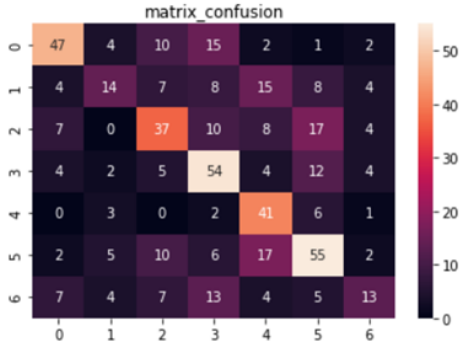
classification

	precision	recall	f1-score	support
0	0.66	0.64	0.65	81
1	0.35	0.47	0.40	60
2	0.53	0.45	0.48	83
3	0.53	0.59	0.56	85
4	0.51	0.75	0.61	53
5	0.68	0.39	0.50	97
6	0.38	0.40	0.39	53
accuracy			0.52	512
macro avg	0.52	0.53	0.51	512
weighted avg	0.54	0.52	0.52	512

0	0.629630
1	0.283333
2	0.530120
3	0.482353
4	0.679245
5	0.474227
6	0.509434

```
Epoch 15/700
110/110 [==============================] - 2s 15ms/step - loss: 0.9556 - accuracy: 0.6655 - val_loss: 0.9446 - val_accuracy:
0.6986
Evaluate on test data
55/55 [==============================] - 0s 6ms/step - loss: 0.9455 - accuracy: 0.6775
```

accuracy: 50.98%



classification

	precision	recall	f1-score	support
0	0.66	0.58	0.62	81
1	0.44	0.23	0.30	60
2	0.49	0.45	0.47	83
3	0.50	0.64	0.56	85
4	0.45	0.77	0.57	53
5	0.53	0.57	0.55	97
6	0.43	0.25	0.31	53
accuracy			0.51	512
macro avg	0.50	0.50	0.48	512
weighted avg	0.51	0.51	0.50	512

```
0    0.716049
1    0.216667
2    0.493976
3    0.623529
4    0.679245
5    0.484536
6    0.339623
```

Dopo aver mostrato i risultati in modo esplicito per uno splitting con n=2 e n=7 nella parte precedente, adesso mostriamo in tabelle riassuntive tutti i risultati raggiunti, al fine di permetterne dei confronti e riuscire a trarre delle conclusioni generali. Gli stessi esperimenti, inoltre, sono stati condotti sulla stessa rete, ma con dataset differenti. In base a quanto illustrato nel capitolo 2, sappiamo che ogni dataset ha delle caratteristiche, che lo contraddistinguono; questo comporta un'estrazione delle features specifica in base al dataset preso in esame ed una sua diversa applicazione sulla rete CNN.

CNN con BERLIN

Tabella 1 - Tabella riassuntiva risultati di Accuracy ottenuti su CNN con Berlin.

Modello	***Maximum Accuracy***	***Minimum Accuracy***	***Overall Accuracy***
CNN			62.30%
CNN con split=2	44.33%	42.19%	43.26%
CNN con split=7	51.17%	46.87%	48.32%

Tabella 2 - Tabella riassuntiva risultati di Accuracy PER CLASS ottenuti su CNN con Berlin.

	class 0	***class 1***	***class 2***	***class 3***	***class 4***	***class 5***	***class 6***
CNN	0.76	0.43	0.53	0.58	0.89	0.62	0.58
CNN con split=2	0.52 0.68	0.37 0.25	0.32 0.38	0.45 0.43	0.58 0.57	0.41 0.57	0.38 0.19
CNN con split=7	0.58 0.53 0.69 0.68 0.58 0.62 0.71	0.30 0.42 0.25 0.18 0.42 0.28 0.22	0.42 0.48 0.58 0.51 0.38 0.53 0.49	0.45 0.43 0.50 0.53 0.48 0.48 0.62	0.72 0.66 0.66 0.58 0.66 0.68 0.68	0.56 0.47 0.35 0.46 0.51 0.47 0.48	0.43 0.40 0.23 0.24 0.41 0.51 0.34

CNN con RAVDESS+SAVEE

Tabella 3 - Tabella riassuntiva risultati di Accuracy ottenuti su CNN con Ravdess + Savee.

Modello	***Maximum Accuracy***	***Minimum Accuracy***	***Overall Accuracy***
CNN			41.77%
CNN con split=2	34.89%	29.36%	32.13%
CNN con split=7	36.90%	31.35%	33.61%

Tabella 4 - Tabella riassuntiva risultati accuracy per class ottenuti su CNN con Ravdess + Savee.

	class 0	***class 1***	***class 2***	***class 3***	***class 4***	***class 5***	***class 6***	***class 7***	***class 8***	***class 9***
CNN	0.31	0.93	0.62	0.42	0.35	0.38	0.78	0.43	0.38	0.54
CNN con split=2	0.70 0.65	0.31 0.50	0.47 0.71	0.33 0.39	0.29 0.35	0.71 0.74	0.61 0.61	0.44 0.40	0.49 0.28	0.48 0.30
CNN con split=7	0.37 0.26 0.37 0.26 0.37 0.37 0.37	0.73 0.91 0.73 1 0.91 0.73 0.64	0.67 0.67 0.33 0.28 0.55 0.50 0.55	0.44 0.28 0.33 0.44 0.28 0.22 0.33	0.50 0.54 0.42 0.75 0.54 0.33 0.42	0.66 0.59 0.59 0.59 0.62 0.62 0.59	0.53 0.53 0.47 0.53 0.47 0.40 0.60	0.40 0.37 0.43 0.33 0.40 0.57 0.40	0.37 0.30 0.44 0.30 0.37 0.33 0.48	0.30 0.33 0.67 0.30 0.45 0.40 0.61

CNN con RAVDESS+TESS

Tabella 5 - Tabella riassuntiva risultati di Accuracy ottenuti su CNN con Ravdess + Tess.

Modello	***Maximum Accuracy***	***Minimum Accuracy***	***Overall Accuracy***
CNN			41.77%
CNN con split=2	37.50%	35.41%	36.45%
CNN con split=7	38.89%	31.59%	34.77%

Tabella 6 - Tabella riassuntiva risultati accuracy per class ottenuti su CNN con Ravdess + Tess.

	class 0	***class 1***	***class 2***	***class 3***	***class 4***	***class 5***	***class 6***	***class 7***
CNN	0.70	0.81	0.38	0.45	0.74	0.25	0.33	0.77
CNN con split=2	0.27 0.45	0.49 0.66	0.71 0.26	0.37 0.42	0.28 0.25	0.43 0.47	0.33 0.37	0.44 0.31
CNN con split=7	0.29 0.27 0.37 0.29 0.39 0.39 0.59 0.44	0.81 0.46 0.33 0.67 0.67 0.53 0.72 0.56	0.38 0.40 0.55 0.28 0.28 0.31 0.33 0.40	0.26 0.23 0.38 0.38 0.38 0.31 0.38 0.33	0.36 0.39 0.40 0.32 0.32 0.50 0.32 0.28	0.53 0.53 0.47 0.41 0.41 0.41 0.53 0.35	0.58 0.38 0.29 0.53 0.53 0.31 0.49 0.31	0.33 0.30 0.57 0.36 0.36 0.36 0.51 0.42

CNN con TESS

Tabella 7 - Tabella riassuntiva risultati di accuracy ottenuti su CNN con Tess.

Modello	***Maximum Accuracy***	***Minimum Accuracy***	***Overall Accuracy***
CNN			100%

Tabella 8 - Tabella riassuntiva risultati di accuracy per class ottenuti su CNN con Tess.

	class 0	***class 1***	***class 2***	***class 3***	***class 4***	***class 5***	***class 6***
CNN	1	1	1	1	1	1	1

Nei paragrafi successivi sono stati discussi i risultati ottenuti e, dopo un confronto attento dei vari valori di accuracy e di loss e delle varie metriche su cui si è agito, sono state tratte delle conclusioni.

3.1.2 Modifiche su rete CNN

Per comprendere meglio cosa influenzi il risultato di accuracy ottenuto da un determinato modello, sono state apportate modifiche di vario tipo su parametri differenti e sulla composizione della rete neurale stessa. La prima modifica che si è deciso di presentare è quella relativa alla rete, infatti è stata aumentata la profondità di rete, aggiungendo altri due livelli di convoluzione, due di Activation e uno di Dropout.

```
model = Sequential()

model.add(Conv1D(256, 5,padding='same',
                 input_shape=(65,1)))
model.add(Activation('relu'))
model.add(Conv1D(128, 5,padding='same'))
model.add(Activation('relu'))
model.add(Dropout(0.1))
model.add(MaxPooling1D(pool_size=(8)))
model.add(Conv1D(128, 5,padding='same',))
model.add(Activation('relu'))
model.add(Conv1D(128, 5,padding='same',))
model.add(Activation('relu'))
model.add(Conv1D(128, 5,padding='same',))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(Conv1D(128, 5,padding='same',))
model.add(Activation('relu'))
model.add(Flatten())
model.add(Dense(7))
model.add(Activation('softmax'))
opt = tf.keras.optimizers.RMSprop(lr=0.00001, decay=1e-6)
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 65, 256)           1536

 activation (Activation)     (None, 65, 256)           0

 conv1d_1 (Conv1D)           (None, 65, 128)           163968

 activation_1 (Activation)   (None, 65, 128)           0

 dropout (Dropout)           (None, 65, 128)           0

 max_pooling1d (MaxPooling1D  (None, 8, 128)           0
 )

 conv1d_2 (Conv1D)           (None, 8, 128)            82048

 activation_2 (Activation)   (None, 8, 128)            0

 conv1d_3 (Conv1D)           (None, 8, 128)            82048

 activation_3 (Activation)   (None, 8, 128)            0

 conv1d_4 (Conv1D)           (None, 8, 128)            82048

 activation_4 (Activation)   (None, 8, 128)            0

 dropout_1 (Dropout)         (None, 8, 128)            0

 conv1d_5 (Conv1D)           (None, 8, 128)            82048

 activation_5 (Activation)   (None, 8, 128)            0

 flatten (Flatten)           (None, 1024)              0

 dense (Dense)               (None, 7)                 7175

 activation_6 (Activation)   (None, 7)                 0

=================================================================
Total params: 500,871
Trainable params: 500,871
Non-trainable params: 0
```

Figura 22 - Rete CNN di partenza con modifica di profondità

Nel fit di questo modello sono stati fissati la batch_size a 16 ed il numero di epoche a 700.

A seguito dell'aumento della profondità della rete neurale e del numero di epoche, si è potuto osservare che il valore di accuracy generale del modello non ha subito grandi variazioni, ma si è mantenuto nello stesso intorno a livello percentuale. Si è verificato un miglioramento, ma non tale da ritenersi rilevante.

In generale, applichiamo architetture di rete più profonde quando sono molti i dati di addestramento etichettati ed il problema di classificazione è impegnativo. Più complessa è la problematica da risolvere, più una rete con un maggior numero di layers sarà in grado di risolverla. La definizione di complessità di un problema è correlata all'Human Error; se l'uomo può risolverlo con una precisione quasi perfetta, il problema non è complesso. Un modello più profondo ha più parametri e più livelli e convoluzioni, e risulta avere prestazioni migliori quando il set di allenamento è più ampio. Il numero di livelli che può essere scelto dipende dai dati, dalla loro dimensione e dal dettaglio dei dati a disposizione. Bisogna stare attenti al fatto che, quanto detto fino ad ora, non significa che più strati si hanno migliore è il risultato ottenuto. L'aggiunta di più livelli aiuta ad estrarre più funzionalità, ma questo può essere fatto entro certi limiti. Dopo un certo limite, invece di estrarre funzionalità, si "sovradattano" i dati e questo comporta irregolarità.

Partiamo dal presupposto che il numero di epoche di allenamento è scelto sulla base dei valori di accuratezza alla fine di ogni epoca, quando questo valore non migliora, si ferma il training. Quello che può capitare è che per un tot di epoche non si registrano miglioramenti nel valore di accuratezza, che rimane comunque positivo; visto che possono verificarsi oscillazioni piccole, diventa importante guardare, non il singolo valore, ma il trend generale del parametro. Guardando superficialmente i dati sembrerebbe essere migliorato il risultato ottenuto, in realtà non è così se si osserva meglio la predizione che viene fatta. Ci si accorge, infatti, che il modello arrivato ad un certo punto non fa altro che overfittare, si verifica un sovradattamento. Quello che è successo è che l'algoritmo ha mostrato relazioni tra un attributo irrilevante ed il risultato finale, quelli trovati sono rumori che sporcano il processo di apprendimento; sono false corrispondenze. L'algoritmo perde di generalità e si adatta troppo ai dati di training.

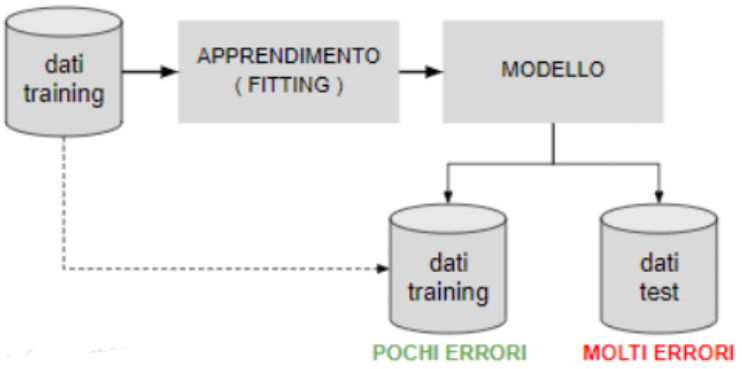


Figura 23 - Overfitting a seguito delle modifiche fatte sul numero di epoche [10]

Per quanto riguarda, invece, la dimensione dell'epoca non influisce sulla precisione, ma è utile per controllare la velocità e le prestazioni. Se la memoria a disposizione è grande, si può avere una dimensione di batch grande e l'allenamento avverrà in un tempo inferiore. Considerazione che può essere fatta è che l'errore sia sul train che sul test si riducono all'aumentare delle epoche. Il fatto che l'accuracy del modello di training sia maggiore dell'accuracy di cross-validation ci fa dedurre che il modello sia andato in overfitting.

3.1.3 Splitting e EarlyStopping

Altre modifiche che possono essere apportate sono quelle relative al set di dati, infatti dataset più ampi permettono di ottenere risultati migliori, questo è possibile notarlo in tutti i test svolti. Bisogna, però, attenzionare oltre che la dimensione del dataset utilizzato, anche la varietà dei dati contenuti. Possono essere apportati diversi accorgimenti per avere migliori risultati e ridurre overfitting o underfitting. Visto che un numero maggiore di epoche porta ad un overfitting ed un numero troppo basso porta ad un underfitting, una soluzione di grande impatto è l'introduzione dell'EarlyStopping.

Partendo da un qualsiasi dataset, prima distinzione da fare, ma non unica, è quella tra training set e testing set. Addestrare una rete e poi testarla sempre con lo stesso testing set, variandone i parametri per cercare la soluzione più efficace ed efficiente al problema che si vuole risolvere, è un errore. Dopo un paio di test si va incontro all'overfitting sul testing set, perché il modello dopo un po' inizia a vedere il testing set come parte del training set. Soluzione a questo scenario sono alcune tecniche di validazione. Dai dataset di partenza sono stati estratti il set di dati per il training, quello per il test ed un terzo set, quello della validation. Un problema è il fatto che andando a fare lo split in tre set, il numero di esempi presenti nel training set si riduce.

Le tecniche di splitting provate sono la K-Fold Cross Validation e la StratifiedKFold.

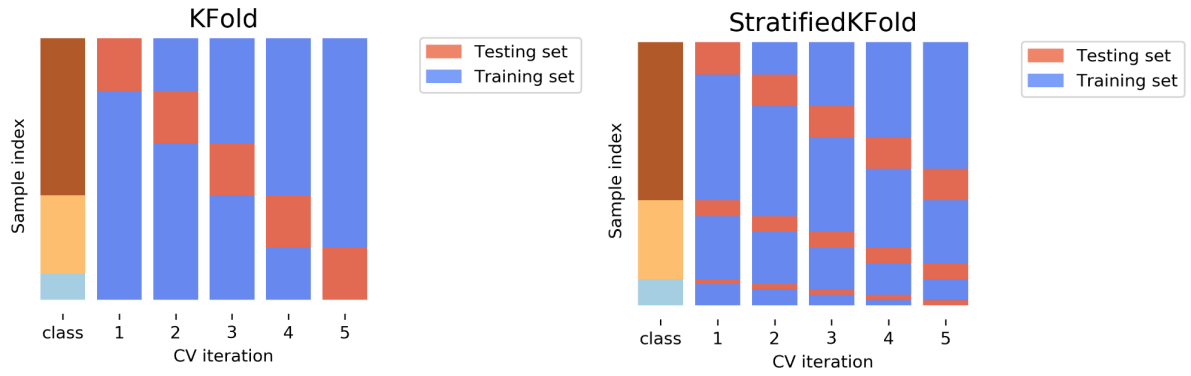


Figura 24 - KFold e StratifiedKFold [11]

La K-Fold Cross Validation fa in modo che il dataset venga diviso in un numero definito di parti e che vengano costruiti lo stesso numero di modelli, la cui valutazione avverrà utilizzando come training set k-1 parti del dataset e come test set una delle k fold. Quello che cambia sono il training set ed il testing set. Calcolando la media dei valori ottenuti a seguito dell'evaluate del modello, si calcola l'errore commesso dal modello nelle predizioni. Per ottenere il risultato migliore è bene tenere il test set solo per testare il modello, in modo che i dati gli siano sconosciuti. Il numero di k folds cambia sulla base delle dimensioni del dataset di partenza. Si può pensare ad un numero di k folds maggiore se il dataset è molto ampio; in generale, k=10 è il valore che si adatta meglio a tutti i dataset. Si è osservato che un numero troppo grande di k fold porta alla condizione di overfitting; mentre un numero troppo basso comporta, spesso, l'underfitting.

Nel nostro caso, è stata utilizzata la StratifiedKFold. Mentre la K-Fold Cross Validation divide il dataset in k fold, StratifiedKFold si assicura che ogni fold del dataset abbia le stesse proporzioni in termini di esempi con le diverse label. Preferiamo questa seconda tecnica perchè il problema che si vuole risolvere in questa tesi è quello relativo alla classificazione di tasks. Le prove che sono state fatte con i vari dataset sui vari modelli, mostrano i diversi risultati ottenuti splittando il dataset in un numero di fold tra 2 e 10.

3.1.4 Modifiche su valori di patience e min_delta

Nel momento in cui viene fatto il fitting del modello, per raggiungere risultati più elevati una scelta effettuabile è quella di introdurre l'EarlyStopping, una funzione che permette di interrompere l'allenamento del modello nel momento in cui tra un'epoca ed un'altra smette di verificarsi un miglioramento della predizione. Osservando l'andamento dell'errore quadratico medio riferito all'estimation sub-set si nota come esso si riduca all'aumentare delle epoche, portando a pensare che la precisione aumenti in maniera inversamente proporzionale ad esse.

Tuttavia, una volta superato il punto di early stopping la rete apprende solamente il rumore contenuto nei dati; quindi, proseguire l'addestramento risulta dannoso, come è già stato affermato. L'EarlyStopping ha come parametri: monitor e patience. Monitor indica quale valore risultante deve essere preso in considerazione, nel nostro caso val_loss e val_accuracy sono le possibilità. Patience indica il numero di epoche senza miglioramento che si devono tollerare, dopo le quali l'addestramento potrà essere interrotto. Le due possibilità di introduzione di EarlyStopping nel fit di un modello sono:

```
callbacks=[EarlyStopping(monitor='val_accuracy', patience=10),
      model_checkpoint_callback])
```

e

```
callbacks=[EarlyStopping(monitor='val_loss', patience=10),
      model_checkpoint_callback])
```

Nel primo esempio viene monitorata la val_accuracy, mentre nel secondo caso la val_loss. Quello che si è potuto notare dai test svolti è che basando il fitting sulla val_loss, i risultati ottenuti sono molto più bassi. Inoltre, si osserva la necessità di un numero maggiore di epoche, al fine di ottenere esiti degni di nota.

In ciascuna epoca i valori calcolati sono la loss e la val_loss e l'accuracy e la val_accuracy. Loss viene applicato al train set, mentre la val_loss al test set. Si presta attenzione alla val_loss, perché questa è una buona indicazione di come il modello si comporti su dati invisibili. È utile la val_loss al fine di prevenire l'overfitting, quest'ultimo si verifica quando la loss continua a diminuire, ma il valore della val_loss è obsoleto o aumenta. Per quanto riguarda la precisione, questa è una metrica solo per la classificazione, in quanto fornisce la percentuale di istanze classificate correttamente. Nelle varie epoche, acc è sui dati di addestramento, mentre val_acc su quelli di convalida. È meglio fare attenzione alla val_acc, perché una rete neurale che finisce per adattare i dati di addestramento al 100%, funzionerebbe male con dati invisibili. Infine, il parametro patience, solitamente, è settato tra 10 e 20. Scegliere un valore di patience piccolo significa che un esperimento si fermerà dopo un intervallo di tempo breve, cioè verrà interrotto anticipatamente; mentre, un valore di patience più grande porterà un esperimento ad attendere per più tempo prima di un'interruzione. Deduciamo che un valore basso comporta miglioramenti dal punto di vista delle tempistiche e a livello di processamento dei dati; ma, al tempo stesso, questo potrebbe comportare uno stop prematuro del fitting della rete. Sappiamo, infatti, che i valori di val_loss e val_accuracy variano continuamente da un'epoca ad un'altra e, quindi, potrebbe verificarsi il caso in cui, settando un valore di patience basso, il processo si conclude ad una precisa epoca, ma con una tolleranza maggiore si sarebbe registrata un'accuracy più elevata nelle epoche successive. È importante settare una tolleranza adeguata, in modo da evitare sottostime del modello.

I risultati ottenuti da un fit basato su val_loss sono da ritenersi non utili, in quanto basando l'accuratezza del modello su questa metrica vengono raggiunti risultati scarsi. Volendo portare un esempio, l'accuracy generale di una rete CNN con split=2 si aggira intorno al 12%, un valore irrisorio.

3.1.5 Normalization dei dati e Scaling

Altri metodi provati per cercare di ottimizzare i modelli e raggiungere risultati migliori sono quelli relativi alla preparazione dei dati.

Lo scaling è un'operazione di pre-processing, utile per il miglioramento degli esiti finali, grazie a questo metodo, in molti casi, si riduce il tempo in cui l'algoritmo di apprendimento converge. Uno degli approcci di scaling più semplice è il ridimensionamento min-max, nel quale vengono individuati il minimo ed il massimo valore tra i valori di una caratteristica, in modo da ridurre le variazioni dei valori. Ogni valore è sostituito con il valore normalizzato ed in questo modo i dati sono resi omogenei.

La normalizzazione è una tecnica usata con l'obiettivo di modifica dei valori delle colonne nel set dei dati, al fine di avere una scala comune senza distorsioni negli intervalli di valori o perdita di informazione. Nello specifico, si è provato ad applicare una normalizzazione nell'intervallo tra 0 e 1

```
# Normalise with MinMaxScaler
from sklearn.preprocessing import MinMaxScaler

# store a copy of the column
names = df3.columns

# create the normaliser
normaliser = MinMaxScaler(feature_range=(0,1))
# the default scale min and max values is 0 and 1

# fit the data to the normaliser
df3_normed = normaliser.fit_transform(df3)

# transform df_normed to a pandas DataFrame
df3_normed = pd.DataFrame(df3_normed, columns = names)

# print head to see that everything works OK
df3_normed.head(15)
```

	0	1	2	3	4	5	6	7	8	9	...	206	207	208	209	210
0	0.123695	0.130471	0.137929	0.138730	0.135135	0.136521	0.136704	0.135970	0.136328	0.138958	...	0.227490	0.245454	0.247275	0.243983	0.234240
1	0.172763	0.182226	0.223754	0.244310	0.242939	0.217790	0.214568	0.294620	0.315387	0.327864	...	0.494215	0.475075	0.476683	0.467593	0.444899
2	0.175185	0.184782	0.195345	0.196478	0.191388	0.193351	0.193610	0.192570	0.193077	0.196801	...	0.612714	0.584916	0.555762	0.536537	0.517506
3	0.185451	0.195610	0.206792	0.207992	0.202603	0.204681	0.204955	0.203855	0.204392	0.191414	...	0.356684	0.359555	0.376518	0.386093	0.423657
4	0.208738	0.247747	0.238694	0.172182	0.125684	0.191008	0.218441	0.192324	0.236077	0.263008	...	0.552552	0.484449	0.460573	0.471975	0.486441
5	0.112714	0.118888	0.125684	0.126413	0.123138	0.124401	0.124568	0.123899	0.124225	0.126621	...	0.416420	0.401623	0.419137	0.434546	0.400973
6	0.221382	0.243001	0.235603	0.223544	0.260415	0.255403	0.234226	0.182839	0.201093	0.262294	...	0.379082	0.331074	0.339760	0.371862	0.336147
7	0.291042	0.295624	0.294643	0.340817	0.353552	0.361495	0.373142	0.355237	0.358665	0.350951	...	0.293047	0.274030	0.281578	0.298722	0.307820
8	0.093967	0.102619	0.111147	0.098194	0.095650	0.096631	0.096760	0.096241	0.096494	0.098356	...	0.213264	0.255157	0.286015	0.267258	0.239177
9	0.338892	0.348524	0.393341	0.394495	0.382583	0.410999	0.406317	0.373882	0.385800	0.407036	...	0.369182	0.368173	0.407459	0.409251	0.380147
10	0.214204	0.248808	0.271479	0.270332	0.267196	0.272258	0.274981	0.273431	0.262210	0.238100	...	0.250873	0.257492	0.249869	0.242264	0.234241
11	0.200480	0.225576	0.265792	0.287518	0.243032	0.250321	0.227921	0.155416	0.151277	0.180226	...	0.141589	0.170559	0.223406	0.219023	0.242451

ed una normalizzazione nell'intervallo tra -1 e 1

```
# Normalise with MinMaxScaler
from sklearn.preprocessing import MinMaxScaler

# store a copy of the column
names = df3.columns

# create the normaliser
normaliser = MinMaxScaler(feature_range=(-1,1))
# the default scale min and max values is 0 and 1

# fit the data to the normaliser
df3_normed = normaliser.fit_transform(df3)

# transform df_normed to a pandas DataFrame
df3_normed = pd.DataFrame(df3_normed, columns = names)

# print head to see that everything works OK
df3_normed.head(15)
```

	0	1	2	3	4	5	6	7	8	9	...	206	207	208	209
0	-0.752610	-0.739058	-0.724141	-0.722541	-0.729729	-0.726957	-0.726592	-0.728060	-0.727343	-0.722084	...	-0.545020	-0.509093	-0.505451	-0.512033
1	-0.654474	-0.635547	-0.552491	-0.511380	-0.514121	-0.564419	-0.570864	-0.410759	-0.369226	-0.344272	...	-0.011569	-0.049851	-0.046634	-0.064813
2	-0.649629	-0.630436	-0.609310	-0.607043	-0.617224	-0.613298	-0.612781	-0.614860	-0.613845	-0.606397	...	0.225427	0.169832	0.111524	0.073074
3	-0.629098	-0.608780	-0.586416	-0.584016	-0.594794	-0.590638	-0.590090	-0.592291	-0.591217	-0.617173	...	-0.286631	-0.280889	-0.246963	-0.227814
4	-0.582525	-0.504506	-0.522613	-0.655636	-0.748633	-0.617985	-0.563118	-0.615353	-0.527846	-0.473985	...	0.105104	-0.031101	-0.078854	-0.056049
5	-0.774573	-0.762224	-0.748632	-0.747173	-0.753724	-0.751198	-0.750865	-0.752202	-0.751550	-0.746758	...	-0.167159	-0.196754	-0.161726	-0.130908
6	-0.557236	-0.513999	-0.528795	-0.552913	-0.479170	-0.489194	-0.531548	-0.634322	-0.597815	-0.475413	...	-0.241836	-0.337851	-0.320481	-0.256277
7	-0.417916	-0.408752	-0.410715	-0.318367	-0.292897	-0.277010	-0.253716	-0.289526	-0.282670	-0.298099	...	-0.413905	-0.451940	-0.436844	-0.402555
8	-0.812067	-0.794762	-0.777707	-0.803612	-0.808700	-0.806738	-0.806479	-0.807518	-0.807011	-0.803289	...	-0.573472	-0.489687	-0.427970	-0.465483
9	-0.322216	-0.302952	-0.213319	-0.211011	-0.234835	-0.178001	-0.187365	-0.252237	-0.228400	-0.185929	...	-0.261636	-0.263654	-0.185082	-0.181497
10	-0.571591	-0.502384	-0.457043	-0.459335	-0.465608	-0.455484	-0.450038	-0.453137	-0.475580	-0.523801	...	-0.498253	-0.485016	-0.500261	-0.515472
11	-0.599041	-0.548848	-0.468415	-0.424965	-0.513936	-0.499358	-0.544159	-0.689168	-0.697446	-0.639549	...	-0.716821	-0.658881	-0.553188	-0.561954

Attraverso un'osservazione dei risultati ottenuti e sopra riportati, è facile notare come l'operazione di normalizzazione nel caso specifico presentato non comporti variazioni di notevole importanza. Con una normalizzazione tra 0 e 1 si raggiunge un'accuracy intorno al 44%, mentre con una normalizzazione nell'intervallo -1 e 1 otteniamo un'accuracy del 40%. In principio, si è pensato di effettuare un'operazione di normalizzazione sui valori di MFCC estratti per eliminare l'influenza di un'eventuale componente rumore; questo perchè i valori di MFCC non sono molto robusti rispetto ad esso. Attraverso lo scaling non otteniamo risultati maggiori di accuracy, ma è possibile notare come si verifichi una riduzione del tempo di convergenza dell'algoritmo al risultato finale. Dalle ricerche condotte è emerso come molti ricercatori per migliorare la robustezza dell'algoritmo MFCC agiscano sull'ampiezza log-mel, infatti tramite un elevamento a potenza di 2 o 3 di tali ampiezze si riesce a ridurre l'influenza di basse componenti energetiche; questo aspetto non è però attenzionato nella ricerca presentata.

3.2 Test su LSTM

A partire dal repository ***aswintechguy/Deep-Learning-Projects***, sono stati fatti dei test su una rete neurale di tipo LSTM. In particolar modo la rete in questione è del tipo:

```
from keras.models import Sequential
from keras.layers import Dense, LSTM, Dropout

model = Sequential([
    LSTM(123, return_sequences=False, input_shape=(40,1)),
    Dense(64, activation='relu'),
    Dropout(0.2),
    Dense(32, activation='relu'),
    Dropout(0.2),
    Dense(7, activation='softmax')
])

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_1 (LSTM)               (None, 123)               61500

 dense_3 (Dense)             (None, 64)                7936

 dropout_2 (Dropout)         (None, 64)                0

 dense_4 (Dense)             (None, 32)                2080

 dropout_3 (Dropout)         (None, 32)                0

 dense_5 (Dense)             (None, 7)                 231

=================================================================
Total params: 71,747
Trainable params: 71,747
Non-trainable params: 0
```

Figura 25 - LSTM model

Nella rete presentata, il primo layer è responsabile dell'acquisizione delle sequenze di input; sono state aggiunte al livello 123 unità, tramite le quali viene calcolato l'output. Nell'input shape del primo livello notiamo i valori (40,1), dove 40 è il numero di samples, ovvero le features. Per migliorare le prestazioni della rete è inserito uno strato Dense ReLU, in questo modo si ottiene una perdita di allenamento inferiore. Quest'ultimo è seguito da un livello Dropout, strato responsabile della regolarizzazione, che impedisce l'overfitting. Il metodo di Dropout, nella fase di addestramento, elimina o ignora un certo numero di neuroni dalla rete; vengono disattivate le attivazioni di alcuni neuroni dello strato LSTM e viene portata avanti una selezione casuale dei nodi con una determinata probabilità (in questo caso del 20%) ad ogni ciclo di aggiornamento del peso. Durante la fase di apprendimento, si stabiliscono i pesi dei neuroni; questi ultimi sono correlati a caratteristiche specifiche, quindi i neuroni vicini si affidano a questa specializzazione, rendendo il modello fragile.

Nel momento in cui alcuni neuroni sono eliminati casualmente, altri neuroni dovranno intervenire e, quindi, ci saranno più interpretazioni indipendenti. La rete diventa meno sensibile ai pesi specifici dei neuroni e, di conseguenza, ci sarà una maggiore generalizzazione; diventa meno probabile che la rete superi i dati di addestramento. Si alternano poi un altro strato Dense e Dropout e, infine, il livello di output calcola la probabilità della previsione. Dopo aver costruito l'architettura del modello, si passa ad addestrarlo.

```
# Train the model
history = model.fit(X_train, y_train, epochs=100, batch_size=512, validation_data=(X_test, y_test))
```

Le epoche scelte in partenza sono 100, ritenute sufficienti per raggiungere buoni risultati. L'intero set di dati deve essere passato attraverso la rete neurale più volte; il dataset è limitato e per ottimizzare il processo di apprendimento si usa il gradiente discendente, che è un processo iterativo. Fatte queste premesse possiamo affermare che non basta una sola epoca per aggiornare i pesi, ma servono più passi. All'aumentare del numero di epoche, più volte il peso viene modificato nella rete neurale e la curva passa da underfitting ad ottimale ad overfitting. Non è possibile definire quale numero di epoche sia da utilizzare, perchè questo numero varia in base al dataset utilizzato e a quanto sono diversi i dati. I set di dati sono divisi in sottoinsiemi, si parla di batch_size, fissata a 512 nel caso specifico. Per completare un'epoca servirà un numero di iterazioni che è uguale al numero di esempi del dataset iniziale diviso la batch_size.

Passiamo, adesso, a descrivere gli step utili per ottenere un buon apprendimento con questa rete neurale.

Una volta scelto il dataset di partenza, vengono calcolati i coefficienti MFCCs.

```
def extract_mfcc(filename):
    y, sr = librosa.load(filename, duration=3, offset=0.5)
    mfcc = np.mean(librosa.feature.mfcc(y=y, sr=sr, n_mfcc=40).T, axis=0)
    return mfcc
```

Nell'estrazione delle features viene specificata la duration in modo che venga considerato l'audio per una quantità di tempo definita, che in questo caso è di 3 secondi. Finito il processo di estrazione delle features, si passa allo splitting dei dati raccolti.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)
```

Preso un set di dati, questo viene suddiviso in due sottoinsiemi, uno è utilizzato per il training, mentre l'altro per il testing, ovvero con esso vengono fatte le previsioni. L'obiettivo di questa suddivisione è quello di utilizzare nuovi dati per stimare le prestazioni del modello. Le percentuali di split utilizzate sono 80% per il train e 20% per il test.

Quando iniziamo la valutazione dei risultati ottenuti, si presta attenzione al validation split per verificare la perdita e l'accuratezza derivanti dal sistema realizzato.

3.2.1 Risultati ottenuti su rete LSTM

Nel primo test svolto con la rete presentata nel paragrafo precedente, si è utilizzato come dataset di partenza TESS, con le sue 200 parole target interpretate con diverse emozioni:

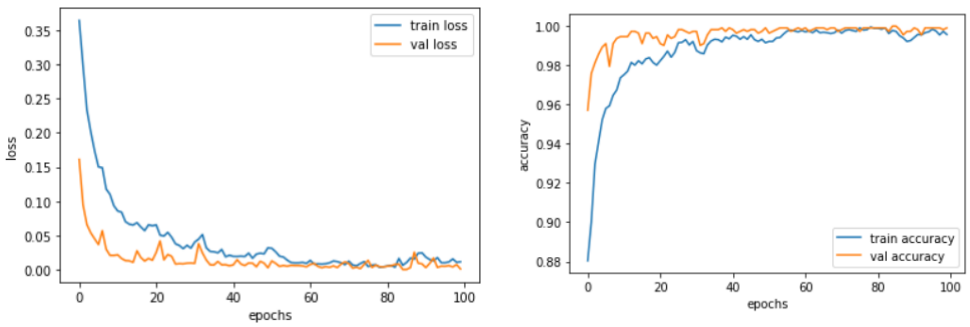
```
angry      800
disgust    800
fear       800
happy      800
neutral    800
ps         800
sad        800
```

A seguito del fit del modello

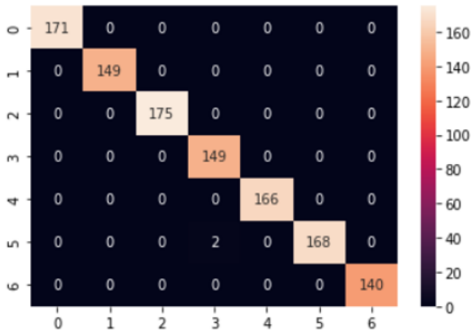
```
# Train the model
history = model.fit(X_train, y_train, epochs=100, batch_size=512, validation_data=(X_test, y_test))
```

vengono raggiunti i seguenti valori:

```
Epoch 98/100
9/9 [==============================] - 5s 511ms/step - loss: 0.0156 - accuracy: 0.9955 - val_loss: 0.0041 - val_accuracy: 0.9
991
Epoch 99/100
9/9 [==============================] - 4s 495ms/step - loss: 0.0107 - accuracy: 0.9973 - val_loss: 0.0073 - val_accuracy: 0.9
982
Epoch 100/100
9/9 [==============================] - 3s 398ms/step - loss: 0.0116 - accuracy: 0.9958 - val_loss: 9.8846e-04 - val_accuracy:
0.9991
```



accuracy del modello: 99.96%



	precision	recall	f1-score	support
0	1.00	1.00	1.00	171
1	1.00	1.00	1.00	149
2	1.00	1.00	1.00	175
3	0.99	1.00	0.99	149
4	1.00	1.00	1.00	166
5	1.00	0.99	0.99	170
6	1.00	1.00	1.00	140
accuracy			1.00	1120
macro avg	1.00	1.00	1.00	1120
weighted avg	1.00	1.00	1.00	1120

0	1.000000
1	1.000000
2	1.000000
3	1.000000
4	1.000000
5	0.987421
6	1.000000

Dal test svolto è possibile concludere dicendo che i risultati ottenuti sono dei migliori, infatti notiamo che il valore di accuracy generale del modello è elevato, arrivando al 99.96%. Andando nello specifico notiamo che le buone prestazioni sono perfettamente rappresentate dalla confusion matrix, dove è raffigurato il perfetto matching delle emozioni tra quelle predette e quelle reali. Infine, anche il classification report è di primaria importanza ai fini della valutazione del riconoscimento vocale delle emozioni svolto. In particolare, grazie alla suddivisione in classi nel report è possibile vedere che l'accuratezza è massima per ogni classe, ad eccezione della classe 5. Nonostante il valore di F1-score non sia massimo, cioè 1, si registra comunque un valore molto alto. Sebbene i risultati soddisfino, sono state apportate alcune modifiche al codice di partenza, per poter definire quali parametri e quali variazioni sui loro valori influenzano il risultato finale di predizione.

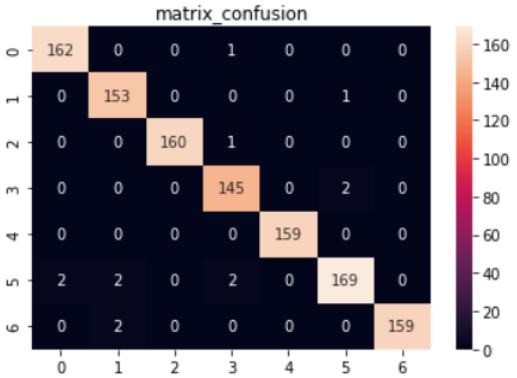
- ***Modifica splitting dati N=2 e introduzione EarlyStopping***

  Oltre allo split del dataset in train set e test set con rapporto 80:20, tramite il metodo StratifiedKFold è stato creato un terzo set, ovvero il validation set. A seguito di questa operazione, segue il calcolo della media dei valori predittivi dei K esperimenti e si seleziona l'albero parziale con il valore predittivo più alto.

  I dati trovati con questa prima modifica sono mostrati di seguito.

```
Epoch 42/700
140/140 [==============================] - 3s 22ms/step - loss: 0.0031 - accuracy: 0.9996 - val_loss: 0.0749 - val_accuracy:
0.9871
Epoch 43/700
140/140 [==============================] - 3s 22ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0619 - val_accuracy:
0.9888
Epoch 44/700
140/140 [==============================] - 3s 23ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0692 - val_accuracy:
0.9884
Evaluate on test data
70/70 [==============================] - 1s 8ms/step - loss: 8.3410e-04 - accuracy: 1.0000
```

accuracy:98.84%



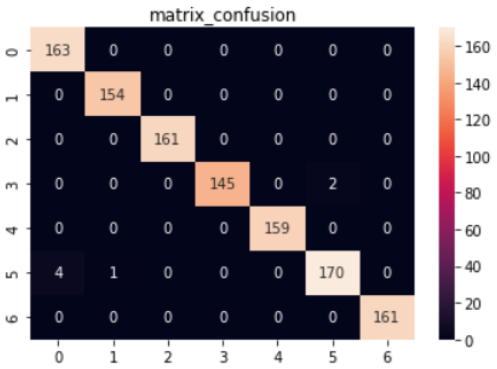
classification

	precision	recall	f1-score	support
0	0.99	0.99	0.99	163
1	0.97	0.99	0.98	154
2	1.00	0.99	1.00	161
3	0.97	0.99	0.98	147
4	1.00	1.00	1.00	159
5	0.98	0.97	0.97	175
6	1.00	0.99	0.99	161
accuracy			0.99	1120
macro avg	0.99	0.99	0.99	1120
weighted avg	0.99	0.99	0.99	1120

	_
0	0.993865
1	0.993506
2	0.993789
3	0.986395
4	1.000000
5	0.965714
6	0.987578

```
Epoch 21/700
140/140 [==============================] - 3s 23ms/step - loss: 4.7814e-04 - accuracy: 1.0000 - val_loss: 0.0040 - val_accura
cy: 0.9991
Epoch 22/700
140/140 [==============================] - 3s 25ms/step - loss: 3.3986e-04 - accuracy: 1.0000 - val_loss: 0.0037 - val_accura
cy: 0.9991
Epoch 23/700
140/140 [==============================] - 3s 23ms/step - loss: 3.9884e-04 - accuracy: 1.0000 - val_loss: 0.0037 - val_accura
cy: 0.9991
Evaluate on test data
70/70 [==============================] - 1s 7ms/step - loss: 2.6098e-04 - accuracy: 1.0000
```

accuracy: 99.37%



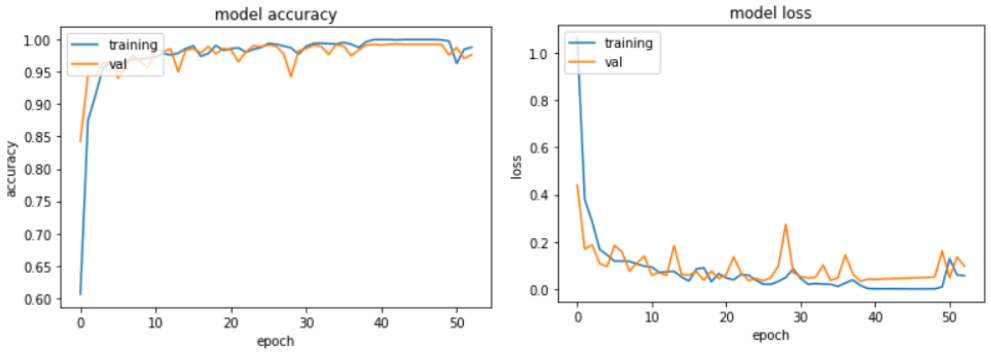
classification

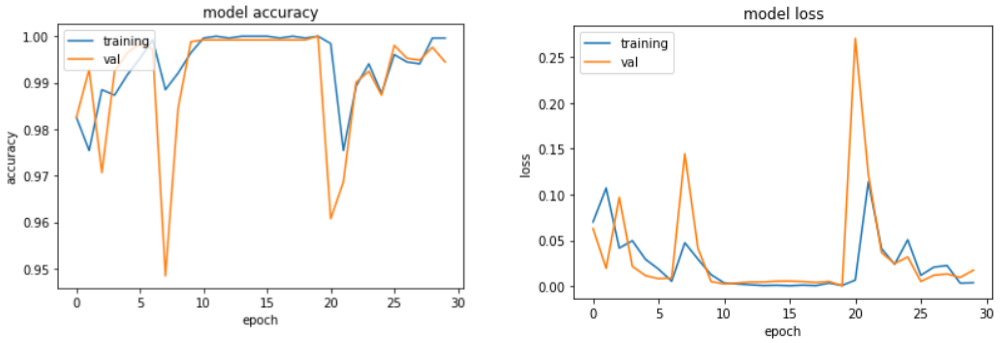
	precision	recall	f1-score	support
0	0.98	1.00	0.99	163
1	0.99	1.00	1.00	154
2	1.00	1.00	1.00	161
3	1.00	0.99	0.99	147
4	1.00	1.00	1.00	159
5	0.99	0.97	0.98	175
6	1.00	1.00	1.00	161
accuracy			0.99	1120
macro avg	0.99	0.99	0.99	1120
weighted avg	0.99	0.99	0.99	1120

0	1.000000
1	1.000000
2	1.000000
3	0.986395
4	1.000000
5	0.971429
6	1.000000

Si osserva che i valori di accuracy, sia quelli generali del modello che quelli di alcune classi, sono diminuiti. Si continuano ad avere ottimi risultati di predizione ma inferiori; questo accade perché si riducono i casi in cui la mancata generalizzazione del modello provoca valutazioni di un certo tipo quando, in realtà, il riferimento sarebbe ad un'altra classe.

Aumentando la dimensione del training e lasciando al test solo, ad esempio, un 10% del dataset di partenza; apparentemente il risultato è migliore in quanto si registra un'accuracy del 100%, ma se si analizzano i plot della val_loss e della val_accuracy, si notano delle oscillazioni nei risultati non indifferenti. Tali oscillazioni testimoniano situazioni di overfitting, il risultato non è attendibile.





Per capire quando le condizioni di overfitting e di underfitting si verificano, inseriamo al seguito un grafico.

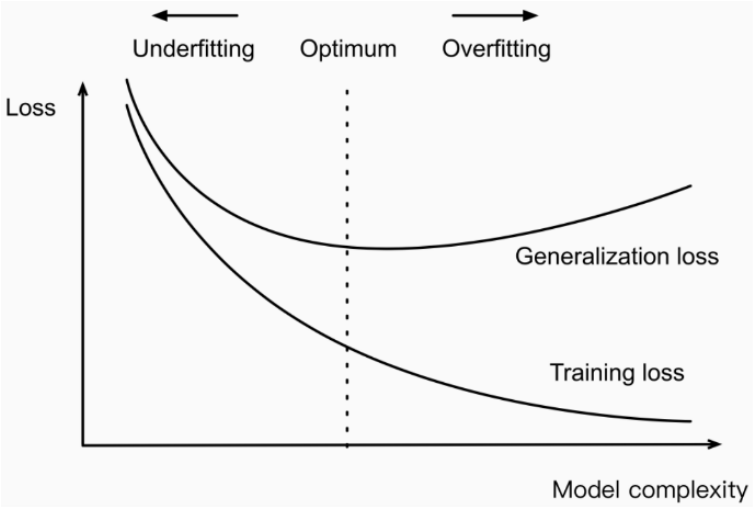
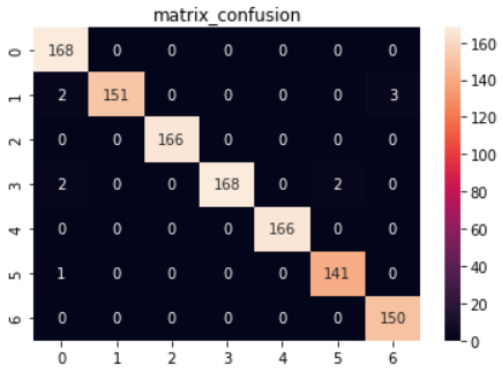


Figura 26 - Condizioni di Overfitting, Optimum e Underfitting di un modello. [12]

- ***Modifica splitting dati N=7 e introduzione EarlyStopping***

```
Epoch 37/700
240/240 [==============================] - 9s 38ms/step - loss: 0.0150 - accuracy: 0.9961 - val_loss: 0.0613 - val_accuracy:
0.9859
Epoch 38/700
240/240 [==============================] - 8s 33ms/step - loss: 0.0308 - accuracy: 0.9924 - val_loss: 0.0383 - val_accuracy:
0.9922
Epoch 39/700
240/240 [==============================] - 10s 41ms/step - loss: 0.0350 - accuracy: 0.9909 - val_loss: 0.1413 - val_accuracy:
0.9781
Evaluate on test data
120/120 [==============================] - 4s 24ms/step - loss: 0.0114 - accuracy: 0.9977
```

accuracy: 99.11%



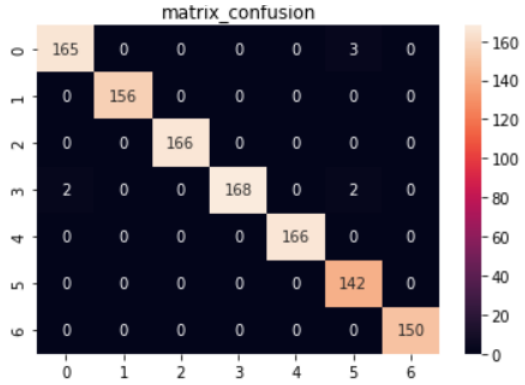
classification

	precision	recall	f1-score	support
0	0.97	1.00	0.99	168
1	1.00	0.97	0.98	156
2	1.00	1.00	1.00	166
3	1.00	0.98	0.99	172
4	1.00	1.00	1.00	166
5	0.99	0.99	0.99	142
6	0.98	1.00	0.99	150
accuracy			0.99	1120
macro avg	0.99	0.99	0.99	1120
weighted avg	0.99	0.99	0.99	1120

0	1.000000
1	0.967949
2	1.000000
3	0.976744
4	1.000000
5	0.992958
6	1.000000

```
Epoch 12/700
240/240 [==============================] - 8s 34ms/step - loss: 0.0158 - accuracy: 0.9961 - val_loss: 0.0209 - val_accuracy:
0.9937
Epoch 13/700
240/240 [==============================] - 9s 39ms/step - loss: 0.0429 - accuracy: 0.9891 - val_loss: 0.0031 - val_accuracy:
1.0000
Evaluate on test data
120/120 [==============================] - 4s 30ms/step - loss: 0.0018 - accuracy: 0.9992
```

accuracy: 99.37%

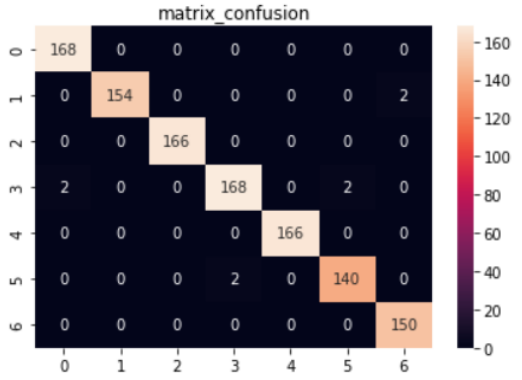


	precision	recall	f1-score	support
0	0.99	0.98	0.99	168
1	1.00	1.00	1.00	156
2	1.00	1.00	1.00	166
3	1.00	0.98	0.99	172
4	1.00	1.00	1.00	166
5	0.97	1.00	0.98	142
6	1.00	1.00	1.00	150
accuracy			0.99	1120
macro avg	0.99	0.99	0.99	1120
weighted avg	0.99	0.99	0.99	1120

0	0.982143
1	1.000000
2	1.000000
3	0.976744
4	1.000000
5	1.000000
6	1.000000

```
Epoch 14/700
240/240 [==============================] - 7s 28ms/step - loss: 7.8496e-04 - accuracy: 1.0000 - val_loss: 9.4154e-05 - val_ac
curacy: 1.0000
Epoch 15/700
240/240 [==============================] - 8s 34ms/step - loss: 6.3660e-04 - accuracy: 1.0000 - val_loss: 4.6533e-05 - val_ac
curacy: 1.0000
Evaluate on test data
120/120 [==============================] - 4s 28ms/step - loss: 2.5534e-04 - accuracy: 1.0000
```

accuracy: 99.28%

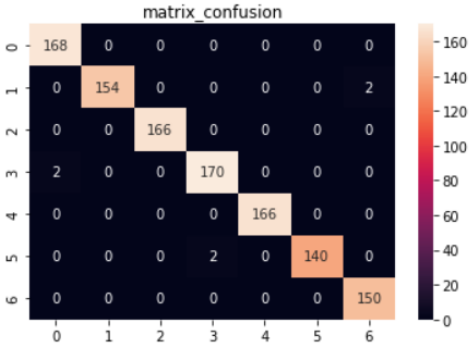


	precision	recall	f1-score	support
0	0.99	1.00	0.99	168
1	1.00	0.99	0.99	156
2	1.00	1.00	1.00	166
3	0.99	0.98	0.98	172
4	1.00	1.00	1.00	166
5	0.99	0.99	0.99	142
6	0.99	1.00	0.99	150
accuracy			0.99	1120
macro avg	0.99	0.99	0.99	1120
weighted avg	0.99	0.99	0.99	1120

0	1.000000
1	0.987179
2	1.000000
3	0.976744
4	1.000000
5	0.985915
6	1.000000

```
Epoch 10/700
240/240 [==============================] - 8s 35ms/step - loss: 0.0607 - accuracy: 0.9846 - val_loss: 0.0131 - val_accuracy:
0.9953
Epoch 11/700
240/240 [==============================] - 9s 36ms/step - loss: 0.0133 - accuracy: 0.9971 - val_loss: 0.0243 - val_accuracy:
0.9937
Epoch 12/700
240/240 [==============================] - 8s 32ms/step - loss: 0.0173 - accuracy: 0.9958 - val_loss: 0.0316 - val_accuracy:
0.9922
Evaluate on test data
120/120 [==============================] - 3s 19ms/step - loss: 0.0011 - accuracy: 1.0000
```

accuracy:99.46%

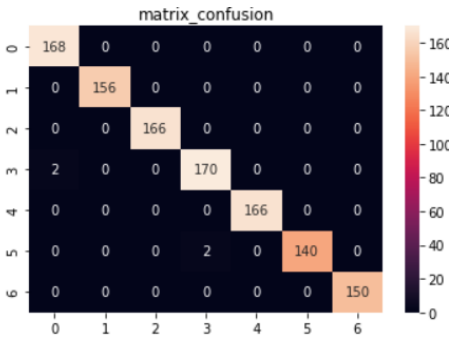


	precision	recall	f1-score	support
0	0.99	1.00	0.99	168
1	1.00	0.99	0.99	156
2	1.00	1.00	1.00	166
3	0.99	0.99	0.99	172
4	1.00	1.00	1.00	166
5	1.00	0.99	0.99	142
6	0.99	1.00	0.99	150
accuracy			0.99	1120
macro avg	0.99	0.99	0.99	1120
weighted avg	0.99	0.99	0.99	1120

0	1.000000
1	0.987179
2	1.000000
3	0.988372
4	1.000000
5	0.985915
6	1.000000

```
Epoch 12/700
240/240 [==============================] - 8s 34ms/step - loss: 0.0259 - accuracy: 0.9937 - val_loss: 0.1442 - val_accuracy:
0.9625
Epoch 13/700
240/240 [==============================] - 8s 32ms/step - loss: 0.0309 - accuracy: 0.9901 - val_loss: 0.0113 - val_accuracy:
0.9969
Epoch 14/700
240/240 [==============================] - 9s 39ms/step - loss: 0.0071 - accuracy: 0.9984 - val_loss: 0.0024 - val_accuracy:
0.9984
Evaluate on test data
120/120 [==============================] - 4s 31ms/step - loss: 2.7290e-04 - accuracy: 1.0000
```

accuracy: 99.64%

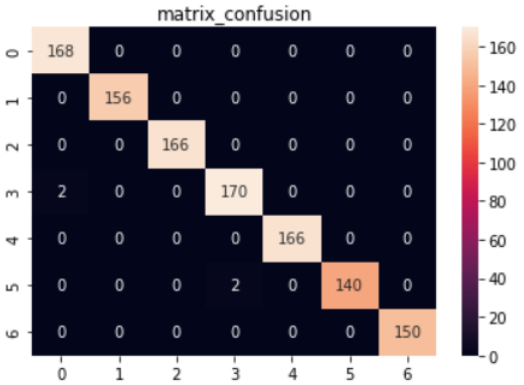


	precision	recall	f1-score	support
0	0.99	1.00	0.99	168
1	1.00	1.00	1.00	156
2	1.00	1.00	1.00	166
3	0.99	0.99	0.99	172
4	1.00	1.00	1.00	166
5	1.00	0.99	0.99	142
6	1.00	1.00	1.00	150
accuracy			1.00	1120
macro avg	1.00	1.00	1.00	1120
weighted avg	1.00	1.00	1.00	1120

0	1.000000
1	1.000000
2	1.000000
3	0.988372
4	1.000000
5	0.985915
6	1.000000

```
Epoch 9/700
240/240 [==============================] - 8s 32ms/step - loss: 0.0178 - accuracy: 0.9961 - val_loss: 0.0028 - val_accuracy:
1.0000
Epoch 10/700
240/240 [==============================] - 7s 27ms/step - loss: 0.0023 - accuracy: 1.0000 - val_loss: 4.7439e-04 - val_accura
cy: 1.0000
Epoch 11/700
240/240 [==============================] - 7s 29ms/step - loss: 7.7044e-04 - accuracy: 1.0000 - val_loss: 2.1300e-04 - val_ac
curacy: 1.0000
Evaluate on test data
120/120 [==============================] - 4s 28ms/step - loss: 1.2049e-04 - accuracy: 1.0000
```

accuracy: 99.64%

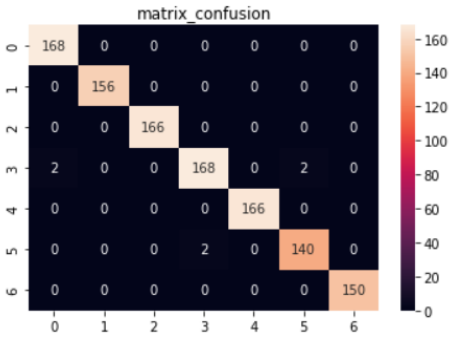


	precision	recall	f1-score	support
0	0.99	1.00	0.99	168
1	1.00	1.00	1.00	156
2	1.00	1.00	1.00	166
3	0.99	0.99	0.99	172
4	1.00	1.00	1.00	166
5	1.00	0.99	0.99	142
6	1.00	1.00	1.00	150
accuracy			1.00	1120
macro avg	1.00	1.00	1.00	1120
weighted avg	1.00	1.00	1.00	1120

```
          _
0    1.000000
1    1.000000
2    1.000000
3    0.988372
4    1.000000
5    0.985915
6    1.000000
```

```
Epoch 9/700
240/240 [==============================] - 7s 30ms/step - loss: 7.7141e-04 - accuracy: 1.0000 - val_loss: 0.0023 - val_accura
cy: 0.9984
Epoch 10/700
240/240 [==============================] - 7s 29ms/step - loss: 3.7011e-04 - accuracy: 1.0000 - val_loss: 0.0025 - val_accura
cy: 0.9984
Epoch 11/700
240/240 [==============================] - 8s 35ms/step - loss: 2.5454e-04 - accuracy: 1.0000 - val_loss: 0.0025 - val_accura
cy: 0.9984
Evaluate on test data
120/120 [==============================] - 4s 25ms/step - loss: 3.4034e-04 - accuracy: 1.0000
```

accuracy: 99.46%



	precision	recall	f1-score	support
0	0.99	1.00	0.99	168
1	1.00	1.00	1.00	156
2	1.00	1.00	1.00	166
3	0.99	0.98	0.98	172
4	1.00	1.00	1.00	166
5	0.99	0.99	0.99	142
6	1.00	1.00	1.00	150
accuracy			0.99	1120
macro avg	0.99	0.99	0.99	1120
weighted avg	0.99	0.99	0.99	1120

Dopo aver mostrato i risultati in modo esplicito nella parte precedente, tutti gli altri risultati sono stati inseriti in tabelle riassuntive, in modo da permetterne i confronti.

LSTM con TESS

Tabella 9 - Tabella riassuntiva risultati di accuracy ottenuti su LSTM con Tess.

Modello	***Maximum Accuracy***	***Minimum Accuracy***	***Overall Accuracy***
LSTM			99.96%
LSTM con split=2	99.37%	98.84%	99.10%
LSTM con split=3	100%	99.55%	99.73%
LSTM con split=4	99.82%	99.28%	99.57%
LSTM con split=5	99.82%	99.37%	99.58%
LSTM con split=6	99.82%	98.75%	98.96%
LSTM con split=7	100%	99.19%	99.59%

Tabella 10 - Tabella riassuntiva risultati di accuracy per class ottenuti su LSTM con Tess.

	class 0	class 1	class 2	class 3	class 4	class 5	class 6
LSTM	1	1	1	1	1	0.99	1
LSTM con split=2	0.97 1	0.99 0.99	1 0.99	0.98 1	1 1	0.99 0.96	1 1
LSTM con split=6	0.98 0.98 0.99 0.98 0.99 1	0.99 0.96 0.99 0.99 1 1	1 1 1 1 1 1	0.99 0.99 1 1 1 1	1 1 1 1 1 1	0.99 0.97 0.99 0.99 0.99 0.99	0.99 1 0.99 1 1 1

LSTM con RAVDESS + SAVEE

Tabella 11 - Tabella riassuntiva risultati di accuracy ottenuti su LSTM con Ravdess + Savee.

Modello	Maximum Accuracy	Minimum Accuracy	Overall Accuracy
LSTM			32.75%
LSTM con split=2	25.40%	25%	25.20%
LSTM con split=7	36.32%	30.04%	33.88%

Tabella 12 - Tabella riassuntiva risultati di accuracy per class su LSTM con Ravdess + Savee.

	class 0	class 1	class 2	class 3	class 4	class 5	class 6	class 7	class 8	class 9
LSTM	0.77	0.88	0.40	0.27	0.84	0.59	0.23	0.26	0.24	0.28
LSTM con split=2	0.29 0.29	0.85 0.89	0.58 0.58	0.37 0.37	0.86 0.86	0.63 0.56	0.33 0.33	0.55 0.48	0.41 0.40	0.38 0.27
LSTM con split=7	0.53 0.87 0.60 0.67 0.73 0.73 0.73	0.62 0.52 0.62 0.76 0.52 0.33 0.42	0.37 0.32 0.37 0.47 0.32 0.32 0.32	0.46 0.54 0.46 0.46 0.54 0.54 0.46	0.88 0.67 0.94 0.94 0.94 0.50 0.78	0.75 0.79 0.63 0.67 0.50 0.79 0.79	0.38 0.33 0.39 0.28 0.28 0.39 0.39	0.36 0.30 0.30 0.47 0.50 0.30 0.33	0.53 0.41 0.47 0.26 0.35 0.44 0.53	0.29 0.29 0.32 0.42 0.35 0.29 0.23

LSTM con RAVDESS + TESS

Tabella 13 - Tabella riassuntiva risultati di accuracy ottenuti su LSTM con Ravdess + Tess.

Modello	***Maximum Accuracy***	***Minimum Accuracy***	***Overall Accuracy***
LSTM			37.50%
LSTM con split=2	28.82%	26.73%	27.77%
LSTM con split=7	36.80%	29.16%	32.88%

Tabella 14 – Tabella risultati di accuracy per class ottenuti su LSTM con Ravdess + Tess.

	class 0	***class 1***	***class 2***	***class 3***	***class 4***	***class 5***	***class 6***	***class 7***
LSTM	0.44	0.45	0.33	0.57	0.71	0.53	0.29	0.39
LSTM con split=2	0.61 0.48	0.71 0.77	0.46 0.38	0.50 0.53	0.28 0.38	0.48 0.78	0.35 0.40	0.66 0.61
LSTM con split=7	0.38 0.31 0.40 0.31 0.47 0.33 0.38	0.78 0.86 0.77 0.69 0.61 0.69 0.81	0.33 0.30 0.27 0.30 0.30 0.24 0.24	0.65 0.58 0.44 0.42 0.39 0.50 0.39	0.29 0.34 0.27 0.24 0.32 0.24 0.19	0.63 0.73 0.53 0.63 0.53 0.58 0.63	0.54 0.62 0.49 0.38 0.38 0.32 0.38	0.57 0.64 0.69 0.50 0.74 0.59 0.67

LSTM con BERLIN

Tabella 15 - Tabella riassuntiva risultati di accuracy ottenuti su LSTM con Berlin.

Modello	***Maximum Accuracy***	***Minimum Accuracy***	***Overall Accuracy***
LSTM			39.84%
LSTM con split=2	45.31%	46.48%	45.90%
LSTM con split=7	54.10%	49.80%	52.45%

Tabella 16 - Tabella riassuntiva risultati di accuracy per class ottenuti su LSTM con Berlin.

	class 0	***class 1***	***class 2***	***class 3***	***class 4***	***class 5***	***class 6***
LSTM	0.54	0.38	0.35	0.27	0.51	0.68	0.57
LSTM con split=2	0.54 0.60	0.42 0.38	0.36 0.24	0.38 0.56	0.57 0.83	0.47 0.42	0.43 0.51
LSTM con split=7	0.54 0.64 0.60 0.54 0.64 0.68 0.69	0.63 0.58 0.40 0.62 0.47 0.37 0.37	0.34 0.42 0.40 0.47 0.41 0.44 0.51	0.49 0.48 0.53 0.55 0.48 0.49 0.46	0.55 0.72 0.74 0.72 0.75 0.77 0.83	0.51 0.42 0.56 0.41 0.55 0.61 0.51	0.47 0.43 0.43 0.40 0.43 0.36 0.51

Nel paragrafo successivo sono stati discussi i risultati ottenuti e, dopo un confronto attento dei vari valori di accuracy e loss e delle varie metriche su cui si è agito, sono state tratte delle conclusioni.

3.2.2 Conclusioni sui test condotti su rete LSTM

Per tracciare delle conclusioni sui test fatti, si è deciso di analizzare i risultati sotto vari aspetti. Come punto di partenza, si possono fare delle considerazioni sui dataset; infatti, ad influire sui risultati è sicuramente la loro composizione. Berlin è il dataset di dimensione più piccola, con 10 speakers che interpretano 7 emozioni diverse, il numero di samples è 700. Ravdess è costituito da 2496 samples e si tratta di 8 emozioni interpretate da 24 persone differenti. Tess è il dataset più vasto con 2800 samples e 7 emozioni interpretate da solamente 2 donne. Savee è un dataset costituito dalle registrazioni di 4 attori maschi in 7 diverse emozioni, sono 480 le espressioni in totale.

Dai test svolti, in questa prima fase, è possibile concludere che i migliori risultati sono ottenuti con Tess. Il motivo è che, un modello che utilizza un dataset con una piccola quantità di dati, non ha molte features su cui basarsi. Inoltre, maggiore è il numero delle classi, più complesso diventa il task di classificazione e più perdita di accuratezza si ottiene.

Le classi con meno accuratezza sono tristezza e sorpresa e neutrale, queste sono le classi più difficili da interpretare, non solo a partire da un audio, ma anche a livello di espressione. Altre due emozioni che vengono spesso confuse sono rabbia e gioia, a causa dei picchi e dell'enfasi che l'uomo presenta quando esprime queste due emozioni.

Altre considerazioni derivano dall'operazione di splitting effettuata sul dataset; anche se in parte tali considerazioni sono legate, per quanto detto sopra, alle caratteristiche dei diversi

dataset. Dai test condotti possiamo affermare che il numero di split scelto influisce non poco sui risultati ottenuti. Introducendo tale operazione, notevoli miglioramenti si registrano sul dataset Berlin, dove si nota che aumentando il numero di split fino ad un massimo di 10, l'accuracy raggiunge percentuali più alte rispetto a quella registrata senza splitting del dataset. Tali miglioramenti non è possibile vederli in modo così netto con gli altri dataset; per i quali, però, notiamo un comportamento comune. In generale, valori di splitting che si adattano bene a qualsiasi dataset sono quelli tra 10 e 20. Questa teoria sembra trovare la sua dimostrazione nei test condotti, infatti notiamo che per dataset come quello derivato dalla combinazione di Ravdess eTess e quello derivato dall'unione di Ravdess e Savee, l'andamento di accuracy è lo stesso. Nei casi in cui lo split viene fatto per un numero di fold da 2 a 7 i valori di accuracy sono inferiori a quelli ottenuti senza introdurre l'operazione di splitting; mentre per valori di fold da 7 fino a 10 l'accuracy registra valori migliori e superiori a quelli del codice di partenza senza splitting.

Concludiamo dicendo che l'andamento delle metriche, in particolare stiamo parlando dell'accuracy dei modelli, è strettamente correlato alla composizione dei dataset con cui vengono condotti i test. Tess è il dataset più ampio ed è anche quello che ci permette di registrare i migliori risultati data, non solo la sua vastità, ma anche la varietà interna che lo contraddistingue. Inoltre, i risultati ottimi ottenuti sono anche da relazionare al fatto che al suo interno sono presenti solo voci femminili.

Nella figura sottostante si mettono a confronto parole identiche pronunciate, in uno stato emozionale di rabbia, da un uomo ed una donna.

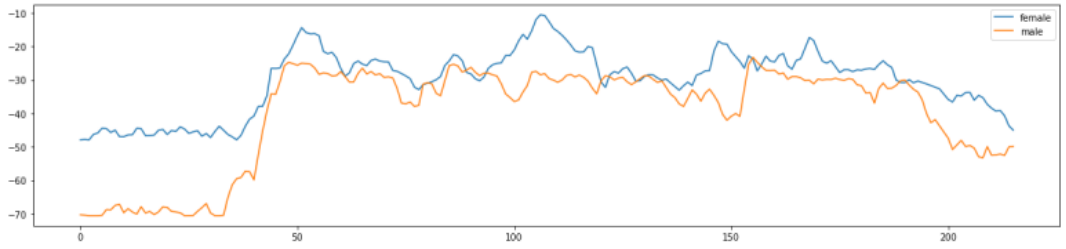


Figura 27 - Rabbia: uomo, donna

In generale, le donne parlano con un tono di voce più alto, circa un'ottava più alta degli uomini; la gamma media di una donna adulta va da 165 a 255 Hz, mentre quella di un uomo va da 85 a 155 Hz. Inoltre, le donne hanno un maggior grado di differenziazione dell'espressività emotiva, sia nella sfera emozionale positiva che su quella negativa; per questo motivo la classificazione emozionale risulta più semplice e comporta una maggiore accuratezza.

3.3 Test su SVM e MLP

Per rendere completa l'analisi dello studio svolto e comprendere l'argomento della tesi fino in fondo ulteriori prove effettuate, sono quelle che coinvolgono l'utilizzo di due delle più importanti reti neurali: MLP e SVM.
Le reti neurali, definibili come approssimatori di dati misurati nello spazio multidimensionale, possono realizzare due diversi tipi di approssimazioni: globale o locale.

Un esempio di rete globale è MLP, Multilayer Perceptron, in cui i neuroni sono disposti in strati; questi ultimi sono quelli di input, output e hidden layer. La rete è di tipo feedforward, ovvero la comunicazione avviene solo tra strati vicini e la propagazione dei segnali di elaborazione avviene dall'ingresso all'uscita. Mentre, un esempio di rete neurale locale è SVM, Support Vector Machine, costituita da uno strato nascosto di unità radiali ed un'uscita neurone.

3.3.1 Test su SVM

I test su SVM sono stati condotti con i dataset Ravdess e Tess, a partire dai quali l'estrazione delle features di interesse è stata svolta attraverso la funzione:

```
def extract_feature(file_name, mfcc):
    result=np.array([])

    X, sample_rate = librosa.load(os.path.join(file_name), res_type='kaiser_fast')
    if mfcc:
        mfccs=np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=40).T, axis=0)
        result=np.hstack((result, mfccs))
    return result
```

Sono stati caricati i file e dopo un ricampionamento ed il calcolo delle MFCCs sono state restituite le features con valore di n_mfcc=40.

A seguito del load dei dati e della distinzione tra attributi ed etichette, il passaggio finale di pre-elaborazione consiste nella suddivisione dei dati in training set e test set; è mantenuta la divisione più usata, ovvero quella con rapporto 80:20.

Partendo dal presupposto che esistono diversi algoritmi SVM, nel caso specifico si è deciso di prendere in considerazione SVC, ovvero il support vector classifier. Tale scelta è giustificata dal fatto che il task da performare è di classificazione. L'obiettivo ultimo di SVC è adattarsi ai dati in input e restituire un iperpiano che li categorizzi. Passiamo, adesso, ad una breve e semplice spiegazione della logica usata da SVM. Alla base di tutto vi è l'idea di creazione di una linea che separi i dati di classi diverse. Immaginando di considerare solo due classi, l'algoritmo ha come obiettivo ultimo quello di trovare una linea che separi due zone dello spazio relativamente ad esse, ovviamente non esiste un'unica linea che separa nello spazio due classi; SVM deve trovare la linea che meglio le separa. Nel Machine Learning bisogna trovare la linea più generale possibile. Secondo l'algoritmo si definiscono i punti più vicini alla

linea da entrambe le classi, questi punti sono detti vettori di supporto. La distanza tra questi vettori e la linea è definita margine, lo scopo è massimizzare il margine. L'iperpiano con margine massimo è quello ottimale.

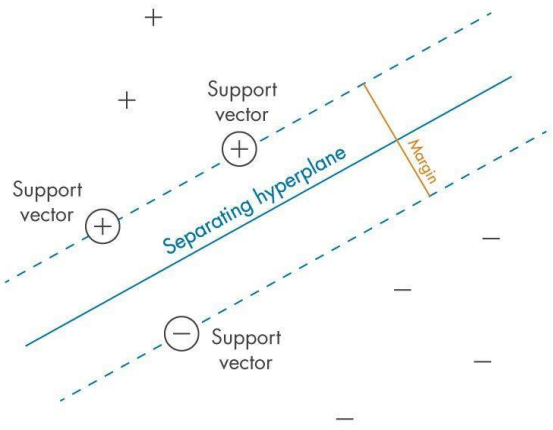


Figura 28 – SVM [13]

Dopo aver ottenuto l'iperpiano, di cui si è parlato in precedenza, si forniscono le features al classificatore per vedere quale classe è stata predetta.

```
from sklearn.svm import SVC

svclassifier = SVC(kernel = 'linear')
```

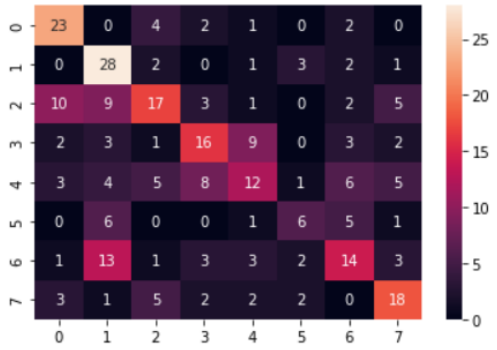
Il kernel è stato settato a "linear". Al dataset è stato applicato uno splitting per distinguere solamente il train set ed il test set. Il fit della classe SVC è stato fatto sui dati di training, al termine di tale operazione è stato applicato il metodo di predizione sui dati di testing.

I risultati ottenuti dal fit della classe SVC

```
svclassifier.fit(X_train, y_train)
```

sono:

```
              precision    recall  f1-score   support

       angry       0.55      0.72      0.62        32
        calm       0.44      0.76      0.55        37
     disgust       0.49      0.36      0.41        47
        fear       0.47      0.44      0.46        36
       happy       0.40      0.27      0.32        44
     neutral       0.43      0.32      0.36        19
         sad       0.41      0.35      0.38        40
   surprised       0.51      0.55      0.53        33

    accuracy                           0.47       288
   macro avg       0.46      0.47      0.46       288
weighted avg       0.46      0.47      0.45       288

----accuracy score 46.52777777777778 ----
```



Infine, calcolando la training accuracy e la testing accuracy si è potuto fare un check sull'overfitting dei dati; a tal proposito, i valori ottenuti sono:

```
train_acc = float(svclassifier.score(X_train, y_train)*100)
print("----train accuracy score %s ----" % train_acc)

test_acc = float(svclassifier.score(X_test, y_test)*100)
print("----test accuracy score %s ----" % test_acc)
```

```
----train accuracy score 63.71527777777778 ----
----test accuracy score 46.52777777777778 ----
```

Per dimostrare la veridicità delle affermazioni fatte nel capitolo 3 sulla base dei vari test svolti, sono state effettuate sul codice, appena presentato, le operazioni di cross-validation e di scaling.

Per quanto concerne la cross-validation si è osservato che, splittando il dataset con valore di n_fold=5, si riesce a raggiungere un valore di accuracy generale del modello intorno al 48% e, quindi, superiore a quello di partenza.

```
from sklearn.model_selection import cross_val_score

# no. of folds cv = 5
cv_results = cross_val_score(svclassifier, X, y, cv = 5)
print(cv_results)
```

```
[0.48958333 0.41319444 0.44791667 0.45486111 0.48263889]
```

Per quanto riguarda la normalizzazione dei dati è stato detto nel capitolo che una semplice normalizzazione sui dati entro un range tra 0 e 1 oppure tra -1 e 1 non permette miglioramenti nei risultati. Era stato, invece, sostenuto che operazioni di scaling differenti possono portare ad incrementi nelle prestazioni del modello. In particolare, si è deciso di provare su questo classifier un metodo di scaling, che consiste nel normalizzare i dati di training sottraendo la media e dividendo per la deviazione standard; quindi, scalare i dati del test con la media e la deviazione standard dei dati di allenamento. Il confronto dei risultati ottenuti è mostrato di seguito.

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.svm import SVC


#splitting dataset into train/ test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)

# Setup the pipeline steps: steps
steps = [('scaler', StandardScaler()),
        ('SVM', SVC())]

# Create the pipeline: pipeline
pipeline = Pipeline(steps)

# Fit the pipeline to the training set: svc_scaled
svc_scaled = pipeline.fit(X_train, y_train)

# Instantiate and fit a classifier to the unscaled data
svc_unscaled = SVC(kernel = 'linear').fit(X_train, y_train)

# Compute and print metrics
print('Accuracy with Scaling: {}'.format(svc_scaled.score(X_test, y_test)))
print('Accuracy without Scaling: {}'.format(svc_unscaled.score(X_test, y_test)))
```

```
Accuracy with Scaling: 0.5694444444444444
Accuracy without Scaling: 0.5277777777777778
```

Effettuando una verifica dell'overfitting o underfitting, confrontando i punteggi di addestramento e test del modello, otteniamo che il discostamento dai risultati ottenuti senza le operazioni di modifica introdotte è di un buon 10%.

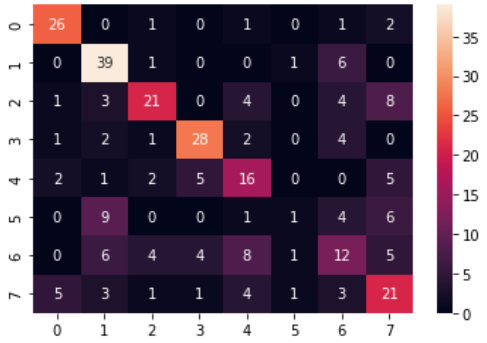
```
train_acc = float(svc_scaled.score(X_train, y_train)*100)
print("----train accuracy score %s ----" % train_acc)

test_acc = float(svc_scaled.score(X_test, y_test)*100)
print("----test accuracy score %s ----" % test_acc)
```

```
----train accuracy score 77.25694444444444 ----
----test accuracy score 56.94444444444444 ----
```

Presentiamo, in definitiva, i risultati ottenuti:

	precision	recall	f1-score	support
angry	0.74	0.84	0.79	31
calm	0.62	0.83	0.71	47
disgust	0.68	0.51	0.58	41
fear	0.74	0.74	0.74	38
happy	0.44	0.52	0.48	31
neutral	0.25	0.05	0.08	21
sad	0.35	0.30	0.32	40
surprised	0.45	0.54	0.49	39
accuracy			0.57	288
macro avg	0.53	0.54	0.52	288
weighted avg	0.55	0.57	0.55	288

----accuracy score 56.94444444444444 ----



A seguito delle varie modifiche apportate notiamo che si è registrato un aumento dell'accuracy generale del modello e che tutte le classi sono state coinvolte in un miglioramento del proprio riconoscimento e della propria classificazione.

3.3.2 Test su MLP

Le reti MLP sono adatte per problemi di classificazione su dati in input a cui viene assegnata una classe o un'etichetta. MLP, ovvero Multilayer Perceptrons, evoca nel suo nome il concetto di perceptron, che è un classificatore lineare. Si tratta di un algoritmo che classifica gli input separandoli in categorie con una linea; il dato in ingresso è, per la maggior parte delle volte, un vettore di features moltiplicate per il loro peso, a cui è aggiunto un bias. Un perceptron produce un unico output. Alla base di MLP troviamo molti perceptron, ognuno dei quali è composto da un layer di input che riceve un dato, un output layer che prende una decisione di predizione e, in mezzo, un numero vario di hidden layers che attuano il processamento dei dati vero e proprio.
La struttura di MLP è mostrata nella figura 29.

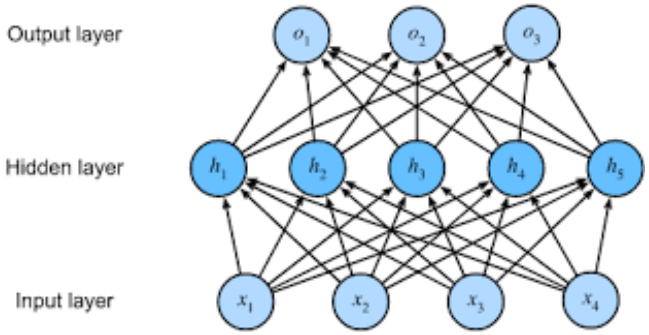


Figura 29 – MLP [14]

I perceptron multistrato vengono spesso applicati a problemi di apprendimento supervisionato: si allenano su un insieme di coppie input-output ed imparano a modellare la relazione tra tali input e output.

Anche i test su MLP sono stati condotti con i dataset Ravdess e Tess, in modo da poter effettuare un confronto diretto tra i due metodi presentati. Questa volta, però, la funzione extract_feature risulta più complessa; oltre alle MFCC features, vengono estratte le caratteristiche chroma e mel.

```
def extract_feature(file_name, mfcc, chroma, mel):
    X, sample_rate = librosa.load(os.path.join(file_name), res_type='kaiser_fast')
    if chroma:
        stft=np.abs(librosa.stft(X))
    result=np.array([])
    if mfcc:
        mfccs=np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=40).T, axis=0)
        result=np.hstack((result, mfccs))
    if chroma:
        chroma=np.mean(librosa.feature.chroma_stft(S=stft, sr=sample_rate).T,axis=0)
        result=np.hstack((result, chroma))
    if mel:
        mel=np.mean(librosa.feature.melspectrogram(X, sr=sample_rate).T,axis=0)
        result=np.hstack((result, mel))
    return result
```

Dopo aver effettuato lo split del dataset, si procede al fit del modello:

```
# Initialize the Multi Layer Perceptron Classifier
model=MLPClassifier(alpha=0.01, batch_size=256, epsilon=1e-08, hidden_layer_sizes=(300,), learning_rate='adaptive', max_iter=500)
```

```
# Train the model
model.fit(x_train,y_train)
```

```
MLPClassifier(alpha=0.01, batch_size=256, hidden_layer_sizes=(300,),
              learning_rate='adaptive', max_iter=500)
```

I risultati ottenuti sono:

```
# Calculate the accuracy of our model
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
# Print the accuracy
print("Accuracy: {:.2f}%".format(accuracy*100))
```

```
Accuracy: 43.33%
```

#classification Report

```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

       angry       0.92      0.43      0.59        51
        calm       0.52      0.38      0.44        45
     disgust       0.49      0.40      0.44        43
     fearful       0.72      0.41      0.52        44
       happy       0.57      0.20      0.30        59
     neutral       0.24      0.77      0.37        31
         sad       0.44      0.36      0.40        45
   surprised       0.35      0.71      0.47        42

    accuracy                           0.43       360
   macro avg       0.53      0.46      0.44       360
weighted avg       0.55      0.43      0.44       360
```

ed infine la Confusion Matrix è

```
from sklearn.metrics import confusion_matrix
matrix = confusion_matrix(y_test,y_pred)
print (matrix)
```

```
[[22  0  5  2  2  5  2 13]
 [ 0 17  5  0  0 19  3  1]
 [ 1  1 17  0  5  9  1  9]
 [ 1  0  4 18  2  7  6  6]
 [ 0  4  2  4 12 14  5 18]
 [ 0  3  0  0  0 24  3  1]
 [ 0  7  0  1  0 13 16  8]
 [ 0  1  2  0  0  9  0 30]]
```

3.3.3 Conclusioni sui test svolti su SVM e MLP

In entrambi i test svolti il dataset utilizzato è lo stesso, costituito da 5252 samples, che rappresentano l'unione di Ravdess e Tess.
Le classi che i modelli vogliono prevedere sono le seguenti: 0 = neutrale, 1 = calma, 2 = gioia, 3 = tristezza, 4 = rabbia, 5 = paura, 6 = disgusto, 7 = sorpresa. Questo set di dati è distorto perché non c'è una classe calma in TESS. Quindi ci sono meno dati per questa emozione ed osservando il rapporto di classificazione se ne vedono le conseguenze. In particolar modo, le predizioni dell'emozione neutrale e della calma risultano errate. Questo perchè, avendo pochi dati sulla calma, il classificatore non riesce a distinguere queste due emozioni con caratteristiche simili. In generale, dall'analisi di quanto ottenuto, possiamo affermare che il modello di classificatore che funziona meglio nel nostro caso specifico è SVM con tutte le operazioni che vi sono state apportate. Il classificatore MLP risulta molto più debole in materia di classificazione delle emozioni.
Riassumiamo i risultati di f1-score per classe ottenuti con questi due classificatori nella tabella sottostante.

Tabella 17 - Risultati per class ottenuti con classificatori SVM e MLP.

class	SVM	MLP
TRISTEZZA	0.32	0.40
RABBIA	0.79	0.59
GIOIA	0.48	0.30
DISGUSTO	0.58	0.44
SORPRESA	0.49	0.47
NEUTRALE	0.08	0.37
CALMA	0.71	0.44
PAURA	0.74	0.52

Per aumentare l'accuratezza e, quindi, migliorare i modelli presentati in questi due ultimi sottoparagrafi si potrebbero raccogliere più dati; in particolare aumentare i dati sulle classi di calma e di emozione neutra. Inoltre, come si è visto, importanti risultati sono stati ottenuti tramite le operazioni svolte nel test su SVM, per cui tali operazioni sono da tenere in considerazione nel momento in cui si vogliono aumentare le prestazioni di un modello di Speech Emotion Recognition.

Applicazione della SER sulla realtà educativa

In questo capitolo tutti gli aspetti trattati fino ad ora, trovano la loro applicazione; infatti, sulla base dei risultati ottenuti dai vari test sono stati introdotti ulteriori esperimenti al fine di completare nel modo più esaustivo possibile la trattazione della presente tesi.
È nel capitolo corrente che verrà raggiunto l'altro importante obiettivo di questa tesi. Se il primo obiettivo era quello di analizzare il mondo dello Speech Emotion Recognition e trarre delle conclusioni sulla base dei risultati ottenuti, al fine di comprendere a pieno come realizzare un modello di predizione il più efficace ed efficiente possibile; l'altro importante scopo è la realizzazione di un codice Python, che permetta la creazione di un modello con possibilità predittive elevate e consenta l'applicazione dello Speech Emotion Recognition sulle videolezioni del Politecnico in modo da muoverci su un livello applicativo.

4.1 Introduzione ad EMOVO

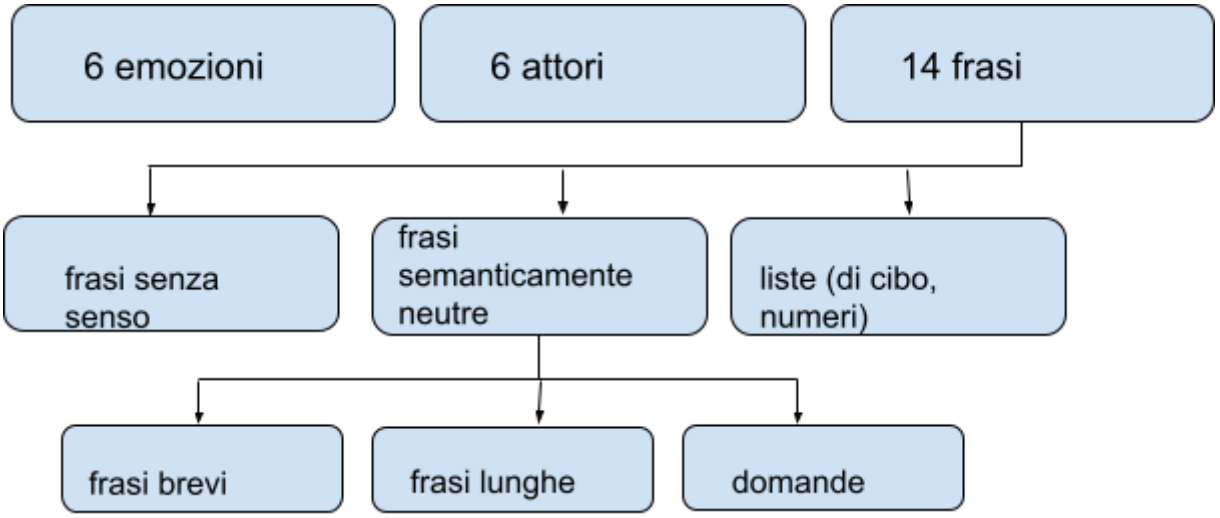


Figura 30 - Struttura EMOVO

Emovo rappresenta il primo corpus di voci emozionali in italiano. Si tratta di un database costruito sulle voci di 6 attori, che interpretano 14 frasi in 6 diversi stati emozionali. Le emozioni sono disgusto, paura, rabbia, gioia, sorpresa, tristezza ed uno stato emozionale neutro. Emovo nasce da registrazioni fatte da professionisti nei laboratori della Fondazione Ugo Bordoni a Roma. Le registrazioni sono state eseguite con due microfoni professionali SHURE SM58LC e con una frequenza di campionamento di 48 kHz, 16 bit in formato PCM stereo wav. In totale si tratta di 588 registrazioni, con durata circa di 60 minuti. Per la validazione di tale dataset vennero organizzati due gruppi di 12 persone in due diversi laboratori, ad ognuno era chiesto di indovinare tra 2 diverse emozioni, al fine di capire quanto le emozioni fossero distinguibili le une dalle altre. Inoltre, per evitare che le persone potessero essere influenzate dal significato semantico delle frasi, vennero fatte ascoltare solo frasi nosense; per un totale di 84 test. I risultati ottenuti dalle interpretazioni sono riportati nella tabella sottostante.

Tabella 18 - Tabella interpretazioni umane avvenuta per la creazione di EMOVO.

Table I		RECOGNIZED EMOTION						
		NEUTRAL	DISGUST	JOY	FEAR	ANGER	SURPRISE	SADNESS
ELICITED EMOTION	NEUTRAL	93%	1%	0%	0%	4%	0%	2%
	DISGUST	3%	67%	2%	6%	10%	6%	6%
	JOY	2%	4%	65%	7%	7%	10%	4%
	FEAR	2%	7%	2%	74%	3%	3%	9%
	ANGER	1%	1%	1%	3%	92%	1%	1%
	SURPRISE	1%	3%	4%	1%	1%	81%	9%
	SADNESS	2%	2%	1%	3%	0%	0%	92%

Come è possibile osservare dai risultati, tutte le classi vennero ben interpretate e distinte con percentuali mai al di sotto del 65%. Le maggiori difficoltà vennero riscontrate per emozioni come il disgusto, scambiato per rabbia, la gioia, scambiata per sorpresa e la paura, scambiata per tristezza. Le emozioni meglio riconosciute, al contrario, risultano essere l'emozione neutrale, la rabbia e la tristezza. Le interpretazioni errate, comunque, trovano motivo di esistere se si considerano tutte le analisi fatte nel Capitolo 2 di questa trattazione. Durante le sessioni di registrazione, gli attori erano liberi di muoversi, questo ha influenzato i valori di intensità assoluta del segnale.

Dal punto di vista sperimentale, il punto di partenza di questo capitolo è l'applicazione di EMOVO sulle reti CNN e LSTM. Presa visione di quale rete funzioni meglio con tale dataset e su quali risultati sia possibile raggiungere apportando tutte le modifiche del caso, si passerà ad applicare il modello di predizione realizzato su audio specifici. Il fine ultimo è fare in modo che la teoria possa trovare un suo campo applicativo nella quotidianità ed in una realtà vicina come quella dell'insegnamento al Politecnico di Torino.

4.2 EMOVO su CNN

La prima applicazione di EMOVO è quella su una **rete CNN** del tipo:

```
model = Sequential()

model.add(Conv1D(256, 5,padding='same',
                 input_shape=(40,1)))
model.add(Activation('relu'))
model.add(Conv1D(128, 5,padding='same'))
model.add(Activation('relu'))
model.add(Dropout(0.1))
model.add(MaxPooling1D(pool_size=(8)))
model.add(Conv1D(128, 5,padding='same',))
model.add(Activation('relu'))
model.add(Conv1D(128, 5,padding='same',))
model.add(Activation('relu'))
model.add(Flatten())
model.add(Dense(7))
model.add(Activation('softmax'))
opt = tf.keras.optimizers.RMSprop(lr=0.00001, decay=1e-6)
```

I dati sono estratti attraverso la funzione:

```
def extract_mfcc(filename):
    y, sr = librosa.load(filename, duration=3, offset=0.5)
    mfcc = np.mean(librosa.feature.mfcc(y=y, sr=sr, n_mfcc=40).T, axis=0)
    return mfcc
```

e poi sono divisi, mantenendo il rapporto 80:20 rispettivamente per il training set e il testing set.

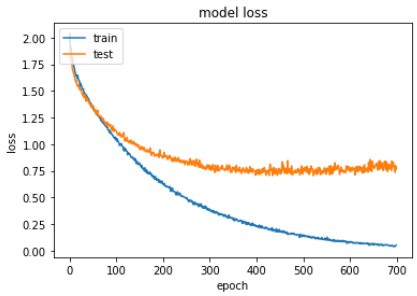
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)
```

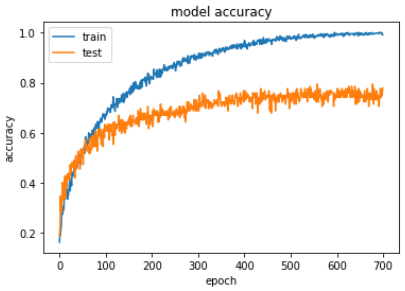
Fatto il fit del modello

```
cnnhistory=model.fit(X_train, y_train, batch_size=16, epochs=700, validation_data=(X_test, y_test))
```

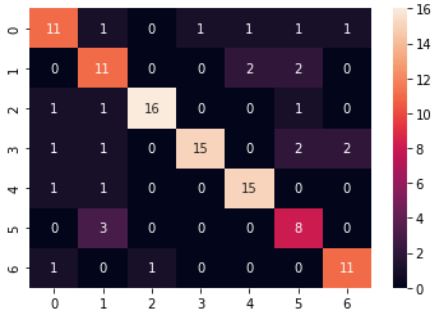
si ottengono i seguenti risultati:

```
Epoch 697/700
28/28 [==============================] - 0s 11ms/step - loss: 0.0389 - accuracy: 1.0000 - val_loss: 0.7999 - val_accuracy: 0.
7411
Epoch 698/700
28/28 [==============================] - 0s 11ms/step - loss: 0.0418 - accuracy: 1.0000 - val_loss: 0.7358 - val_accuracy: 0.
7768
Epoch 699/700
28/28 [==============================] - 0s 12ms/step - loss: 0.0454 - accuracy: 0.9910 - val_loss: 0.7942 - val_accuracy: 0.
7500
Epoch 700/700
28/28 [==============================] - 0s 12ms/step - loss: 0.0518 - accuracy: 0.9910 - val_loss: 0.7663 - val_accuracy: 0.
7768
```





L'accuracy generale del modello è del 77.68%.



	precision	recall	f1-score	support
0	0.73	0.69	0.71	16
1	0.61	0.73	0.67	15
2	0.94	0.84	0.89	19
3	0.94	0.71	0.81	21
4	0.83	0.88	0.86	17
5	0.57	0.73	0.64	11
6	0.79	0.85	0.81	13
accuracy			0.78	112
macro avg	0.77	0.78	0.77	112
weighted avg	0.80	0.78	0.78	112

	row_0
0	0.687500
1	0.733333
2	0.842105
3	0.714286
4	0.882353
5	0.727273
6	0.846154

dtype: float64

Primo importante risultato di questo paragrafo è il raggiungimento di un valore di accuracy generale del modello del 77.68%. Le classi meglio predette sono la 2 e la 6; mentre quelle che presentano più interpretazioni errate sono la classe 0 e la 3.

Secondo la trattazione condotta nel Capitolo 3, si è deciso di apportare delle modifiche al codice di partenza del modello di apprendimento appena mostrato, per avere un modello il più accurato possibile nelle predizioni.

Prima operazione effettuata è stata quella di splitting, i risultati ottenuti dai vari test condotti sono stati riassunti nella tabella presente nella pagina successiva. Importante considerazione da fare è che, tra i test condotti, sono stati inseriti in tabella quelli che hanno portato a risultati più elevati.

CNN con EMOVO

Tabella 19 - Tabella riassuntiva risultati di Accuracy ottenuti su CNN con EMOVO.

Modello	***Maximum Accuracy***	***Minimum Accuracy***	***Overall Accuracy***
CNN			77.68%
CNN con split=2	40.18%	33.03%	36.60%
CNN con split=7	45.54%	41.07%	42.47%
CNN con split=10	48.21%	39.29%	31.51%
CNN con split=15	56.25%	24.10%	37.67%
CNN con split=20	61.60%	33.03%	46.47%

Apportando modifiche di normalizzazione e di splitting dei dati, in modo da avere training, validation e testing set è stato possibile raggiungere i seguenti risultati.

Tabella 20 - Tabella risultati di Accuracy con splitting e normalization su CNN con EMOVO.

Modello	***Maximum Accuracy***	***Minimum Accuracy***	***Overall Accuracy***
CNN			60.71%
CNN con split=2	29.80%	28.84%	29.32%
CNN con split=7	19.35%	16.12%	17.85%
CNN con split=10	28.15%	21.35%	24.17%

4.2.1 Commento ai risultati trovati

In questo paragrafo ci si è proposti come obiettivo l'analisi di tutti i dati ottenuti dai diversi test svolti con EMOVO su CNN. Come punto di partenza l'accuracy generale del modello è di 77.68%, il fit del modello è stato condotto per 700 epoche; ma si è osservato che sono sufficienti già 400 epoche per avere una condizione di stazionarietà nei valori di val_loss e val_accuracy. Dopo circa 400 epoche si registrano oscillazioni delle metriche, ma sempre nello stesso intorno di valori, il modello ha già definito una regola generale per il problema posto. A livello di predizioni osserviamo che tutte le classi vengono ben predette, i valori più elevati si registrano per le classi 2, 3 e 4; mentre quelli più bassi per le classi 1 e 5.

Passando, poi, all'analisi dei dati prodotti dall'aggiunta della procedura di splitting è facile osservare che, per valori n_fold elevati, i valori di accuracy e di loss tendono ad avere un andamento molto oscillante. In questi casi i risultati ottenuti non permettono di raggiungere condizioni sufficienti per la realizzazione di un modello ottimale. A seguito del fit del modello ci sono alcune classi che non riescono ad essere predette correttamente, in particolar modo la classe 5, la maggior parte delle volte, non viene affatto riconosciuta e presenta un f1-score dello 0% o estremamente basso. Le difficoltà appena mostrate, si riscontrano anche per altri valori di n_fold elevati. Al contrario, per valori di n_fold=2 la predizione presenta una situazione migliore. Dagli esperimenti ci accorgiamo che, la procedura di splitting non dovrebbe essere utilizzata quando si dispone di un set di dati piccolo oppure quando il set di dati non è bilanciato o bilanciabile. Per poter ottenere dei risultati concreti con questa operazione devono esserci record sufficienti a coprire tutti i casi comuni e i casi più rari nel dominio problematico. Se il set di dati è troppo piccolo, dividere in train e test, significa non avere dati sufficienti per consentire un buon apprendimento al modello ed avere una buona mappatura degli input sugli output. La stima del modello potrebbe risultare o eccessivamente ottimistica o eccessivamente pessimistica. Sicuramente in termini computazionali lo splitting ci permette di avere costi inferiori, ma a livello di risultati ottenibili non abbiamo valori degni di nota.

Prima di passare all'analisi dei dati ottenuti dall'introduzione della normalizzazione sulle features estratte, usate all'interno del modello, partiamo dal presupposto che quando i dati presentano scale variabili e l'algoritmo che si sta utilizzando non fa ipotesi sulla loro distribuzione diventa utile la normalizzazione. La normalizzazione che si è scelta è quella nell'intervallo di valori tra 0 e 1. Per risolvere la sfida dell'apprendimento del modello vengono normalizzati i dati, assicurandoci che le varie caratteristiche abbiano intervalli di valori simili (ridimensionamento delle funzioni), in modo che le discese del gradiente possano convergere più velocemente. A seguito di tale operazione notiamo che la miglior risposta del modello è quella data da CNN senza splitting. Dagli altri test è emerso che, attuando la normalizzazione e lo splitting, non si raggiungevano valori soddisfacenti; in particolare, molte classi non riuscivano ad essere predette dal modello e presentavano valori di precision e di f1-score dello 0%.

A seguito delle considerazioni e delle analisi svolte, si ritiene che modelli da tenere in considerazione possano essere quelli con la rete CNN senza splitting sia con che senza normalization e quello con splitting con n_fold=2 senza normalization.

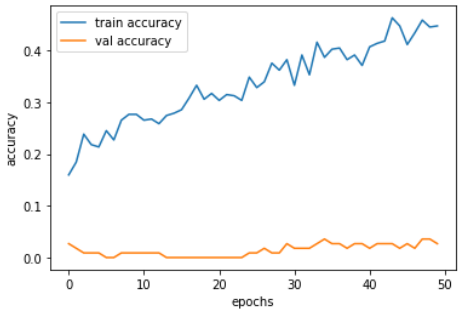
4.3 EMOVO su LSTM

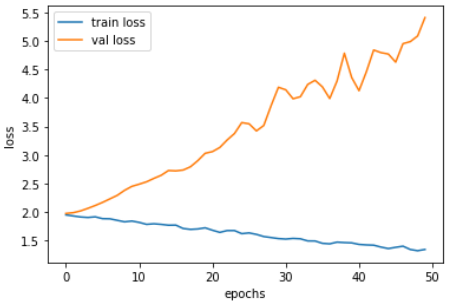
La seconda applicazione di EMOVO è quella su una rete **LSTM** del tipo:

```
from keras.models import Sequential
from keras.layers import Dense, LSTM, Dropout

model = Sequential([
    LSTM(128, return_sequences=False, input_shape=(40,1)),
    Dense(64, activation='relu'),
    Dropout(0.4),
    Dense(32, activation='relu'),
    Dropout(0.4),
    Dense(7, activation='softmax')
])

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()
```

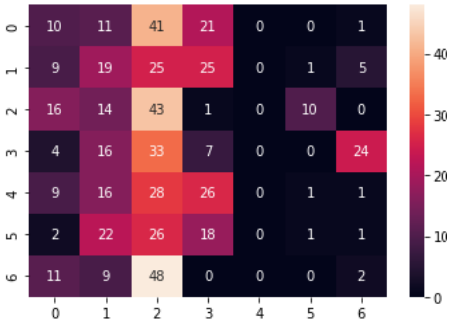
L'estrazione dei dati ed il loro splitting è mantenuto uguale a quello mostrato nel paragrafo 4.1. Avviando il fit del modello, i risultati ottenuti sono riportati al seguito:

```
Epoch 46/50
2/2 [==============================] - 1s 517ms/step - loss: 1.3812 - accuracy: 0.4112 - val_loss: 4.6260 - val_accuracy: 0.0
268
Epoch 47/50
2/2 [==============================] - 1s 391ms/step - loss: 1.4012 - accuracy: 0.4337 - val_loss: 4.9504 - val_accuracy: 0.0
179
Epoch 48/50
2/2 [==============================] - 1s 491ms/step - loss: 1.3427 - accuracy: 0.4584 - val_loss: 4.9863 - val_accuracy: 0.0
357
Epoch 49/50
2/2 [==============================] - 1s 354ms/step - loss: 1.3210 - accuracy: 0.4449 - val_loss: 5.0856 - val_accuracy: 0.0
357
Epoch 50/50
2/2 [==============================] - 1s 483ms/step - loss: 1.3437 - accuracy: 0.4472 - val_loss: 5.4072 - val_accuracy: 0.0
268
```





L'accuracy generale del modello è del 42.19%.



	precision	recall	f1-score	support
0	0.16	0.12	0.14	84
1	0.18	0.23	0.20	84
2	0.18	0.51	0.26	84
3	0.07	0.08	0.08	84
4	0.00	0.00	0.00	81
5	0.08	0.01	0.02	70
6	0.06	0.03	0.04	70
accuracy			0.15	557
macro avg	0.10	0.14	0.11	557
weighted avg	0.11	0.15	0.11	557

Il primo risultato di questo paragrafo è il raggiungimento di un valore di accuracy generale del modello del 42.19%. La predizione delle classi non è per nulla soddisfacente. A partire, già, dalla rappresentazione visiva della Confusion Matrix di questo modello notiamo che le predizioni delle classi 3, 4, 5 e 6, non raggiungono valori né minimi né sufficienti per poter parlare di convergenza del modello o di predizione emozionale. Nonostante i valori ottenuti, secondo quanto descritto ed analizzato nel Capitolo 3, si è deciso di vedere se apportando delle modifiche, si potessero ottenere esiti diversi. Anche in questo caso, la prima operazione effettuata è stata quella di splitting, i risultati ottenuti dai vari test condotti sono stati riassunti nella tabella al seguito.

LSTM con EMOVO

Tabella 21 - Tabella riassuntiva risultati di accuracy ottenuti su LSTM con EMOVO.

Modello	***Maximum Accuracy***	***Minimum Accuracy***	***Overall Accuracy***
LSTM			42.19%
LSTM con split=2	54.46%	51.78%	53.12%
LSTM con split=7	62.50%	44.64%	55.09%
LSTM con split=10	66.07%	54.46%	60.35%
LSTM con split=15	56.25%	35.71%	47.61%
LSTM con split=20	58.03%	45.53%	51.95%

Apportando modifiche di normalizzazione e di splitting dei dati, in modo da avere training, validation e testing set è stato possibile raggiungere i seguenti risultati.

Tabella 22 - Tabella risultati di accuracy con splitting e normalization su LSTM con EMOVO.

Modello	***Maximum Accuracy***	***Minimum Accuracy***	***Overall Accuracy***
LSTM con split=2	15.83%	14.16%	14.99%
LSTM con split=7	41.07%	33.03%	40.03%
LSTM con split=10	44.66%	16.50%	38.05%

4.3.1 Commento ai risultati trovati

In questo paragrafo ci si è posti come obiettivo l'analisi di tutti i dati ottenuti dai diversi test svolti con EMOVO su LSTM. Come punto di partenza l'accuracy generale del modello è di 42.19%. Il fit del modello è stato condotto per 50 epoche, notiamo che il modello non riesce a generalizzare una regola di predizione per alcune classi come le classi 4, 5 e 6. Per le classi appena citate il valore di f1-score rimane sotto valori dello 0.04. Le altre classi presentano valori più elevati accuracy, ma comunque l'errore che il modello compie sulle predizioni è estremamente elevato; a tal punto da non poter considerare questa soluzione applicabile per ottenere buoni risultati al fine dell'obiettivo di questa tesi. Passando, poi, all'analisi dei dati prodotti dall'aggiunta della procedura di splitting è facile osservare che le migliori predizioni sono quelle che si ottengono dal test condotto sulla rete LSTM con splitting con n_fold=10. In tale test si osserva un'accuracy media generale del 60.35%, con accuracy maggiore del 66.07% e minore del 54.46%. Per concludere, introducendo come modifica aggiuntiva quella relativa alla normalizzazione dei dati, notiamo che la miglior risposta del modello è quella che si ottiene con la rete LSTM con splitting con n_split=7.

4.4 Creazione modello con rete CNN per EMOVO

Il dataset EMOVO è stato applicato su una rete CNN per effettuare il riconoscimento delle emozioni su lezioni in italiano, tenutesi al Politecnico di Torino. Nello specifico, le lezioni analizzate sono quelle di tre materie diverse: lezioni di Elettrotecnica, di Sistemi elettronici: tecnologie e misure, di Comunicazione multimediale.

A seguito delle considerazioni fatte e da un'attenta analisi delle metriche derivate dal fit dei vari modelli si ritiene che, per la realizzazione del modello di nostro interesse, possa essere sfruttata la rete CNN. Mentre, per quanto riguarda i modelli con rete LSTM, osserviamo che sono rari i casi di convergenza ottenuti e che i dati ottenuti non sono sufficienti a permetterci delle predizioni delle emozioni vicine alla realtà. Sono molti i casi in cui il valore di val_accuracy diminuisce, mentre il valore di val_loss aumenta; questo significa che il modello, per lo più, si riempie di valori, ma senza imparare nulla.

Il modello di apprendimento costituito da una rete neurale di tipo CNN, derivato dallo studio condotto e presentato in questa tesi, è il seguente:

```
import pandas as pd
import librosa
import librosa.display
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from matplotlib.pyplot import specgram
import keras
from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Dense, Embedding
from keras.layers import LSTM
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.utils import to_categorical
from keras.layers import Input, Flatten, Dropout, Activation
from keras.layers import Conv1D, MaxPooling1D, AveragePooling1D
from keras.models import Model
from keras.callbacks import ModelCheckpoint
from sklearn.metrics import confusion_matrix
from tensorflow.keras import optimizers
from matplotlib import pyplot as plt
import IPython.display as ipd
import librosa
import pandas as pd
import os
import seaborn as sns
from IPython.display import Audio
from sklearn.metrics import classification_report
import warnings
warnings.filterwarnings('ignore')
```

Una volta importati i moduli e le librerie fondamentali per il funzionamento di tutto il codice Python presentato, si passa al caricamento del dataset italiano, EMOVO, usato per l'apprendimento del modello e poi per effettuare le predizioni sulla base dei dati ottenuti.

```
paths = []
labels = []
for dirname, _, filenames in os.walk('./EMOVO'):
    for filename in filenames:
        paths.append(os.path.join(dirname, filename))
        label = filename.split('-')[0]
        label = label.split('.')[0]
        labels.append(label.lower())
    if len(paths) == 2800:
        break
print('Dataset is Loaded')
```

```
Dataset is Loaded
```

```
paths[:5]
```

```
['./EMOVO\\OAF_angry\\rab-m1-b1.wav',
 './EMOVO\\OAF_angry\\rab-m1-b2.wav',
 './EMOVO\\OAF_angry\\rab-m1-b3.wav',
 './EMOVO\\OAF_angry\\rab-m1-d1.wav',
 './EMOVO\\OAF_angry\\rab-m1-d2.wav']
```

```
labels[:5]
```

```
['rab', 'rab', 'rab', 'rab', 'rab']
```

Il dataset è costituito da 84 tracce per il disgusto, la paura, la gioia, l'emozione neutra; da 81 tracce per la rabbia; 70 tracce per la sorpresa e la tristezza.

```
df['label'].value_counts()
```

```
dis    84
pau    84
gio    84
neu    84
rab    81
sor    70
tri    70
Name: label, dtype: int64
```

```
df['label']
```

```
0      rab
1      rab
2      rab
3      rab
4      rab
      ...
552    tri
553    tri
554    tri
555    tri
556    tri
Name: label, Length: 557, dtype: object
```

In questa prima parte di codice sono state distinte le label relative alle emozioni e, successivamente si è passati all'estrazione delle features utili al fine dell'apprendimento del modello. Dopo l'operazione di load per i file audio di riferimento, è stata effettuata l'estrazione delle componenti mfcc.

```
def extract_mfcc(filename):
    y, sr = librosa.load(filename, duration=3, offset=0.5)
    mfcc = np.mean(librosa.feature.mfcc(y=y, sr=sr, n_mfcc=40).T, axis=0)
    return mfcc
```

```
extract_mfcc(df['speech'][0])
```

```
array([-1.48629440e+02,  5.36020851e+01, -1.98867664e+01,  2.19076557e+01,
       -9.28205299e+00, -1.31412525e+01, -2.49307880e+01,  2.95496559e+00,
       -3.21160054e+00,  3.47400498e+00, -5.68868160e+00,  6.42438889e+00,
       -9.06921005e+00, -2.13667059e+00, -6.24426508e+00, -9.46849287e-01,
       -1.28875456e+01, -6.38718963e-01, -9.04429722e+00,  5.70776522e-01,
       -8.68056107e+00, -5.91435075e-01, -1.03111010e+01, -2.25432396e+00,
       -4.96219158e+00,  6.90365851e-01, -5.62512636e+00,  5.31791270e-01,
       -4.46216774e+00, -5.68572044e-01, -4.05416298e+00,  3.53224826e+00,
       -3.67620736e-01,  3.47265649e+00, -1.41735539e-01,  9.24442768e-01,
       -1.04648876e+00,  9.55286622e-01,  1.50571346e-01,  7.47854531e-01],
      dtype=float32)
```

```
X_mfcc = df['speech'].apply(lambda x: extract_mfcc(x))
```

```
X_mfcc
```

```
0      [-148.62944, 53.602085, -19.886766, 21.907656,...
1      [-170.65163, 54.795918, -15.965635, 30.213524,...
2      [-188.35916, 73.04902, -43.614635, 27.211641, ...
3      [-177.11932, 73.996216, -55.287792, 9.002961, ...
4      [-400.27933, 68.093575, -13.613334, 8.989099, ...
                             ...
552    [-504.5839, 71.02924, 24.830746, 35.03621, 10....
553    [-530.1834, 64.49468, 25.985163, 31.781223, 9....
554    [-556.3749, 73.846954, 29.450531, 43.050858, 1...
555    [-577.1512, 64.716644, 26.517622, 30.594944, 9...
556    [-623.86646, 67.75198, 36.253944, 31.5974, 10....
Name: speech, Length: 557, dtype: object
```

```
X = [x for x in X_mfcc]
X = np.array(X)
X.shape
```

```
(557, 40)
```

```
## input split
X = np.expand_dims(X, -1)
X.shape
```

```
(557, 40, 1)
```

```
from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder()
y = enc.fit_transform(df[['label']])
```

```
y = y.toarray()
```

```
y.shape
```

```
(557, 7)
```

Il modello creato è di tipo sequenziale, in quanto vengono immesse e generate sequenze di dati; nel caso specifico si considerano clip audio come esempi. La rete CNN è costituita da 4 livelli convoluzionali, 5 livelli con funzione di attivazione, un livello con Dropout, uno di MaxPooling1D, uno di tipo Dense ed uno in cui è inserita la funzione Flatten. L'ottimizzatore usato è RMS.

```
#Create model
```

```
model = Sequential()

model.add(Conv1D(256, 5,padding='same',
                 input_shape=(40,1)))
model.add(Activation('relu'))
model.add(Conv1D(128, 5,padding='same'))
model.add(Activation('relu'))
model.add(Dropout(0.1))
model.add(MaxPooling1D(pool_size=(8)))
model.add(Conv1D(128, 5,padding='same',))
model.add(Activation('relu'))
model.add(Conv1D(128, 5,padding='same',))
model.add(Activation('relu'))
model.add(Flatten())
model.add(Dense(7))
model.add(Activation('softmax'))
opt = tf.keras.optimizers.RMSprop(lr=0.00001, decay=1e-6)
```

```
model.summary()
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 40, 256)           1536

 activation (Activation)     (None, 40, 256)           0

 conv1d_1 (Conv1D)           (None, 40, 128)           163968

 activation_1 (Activation)   (None, 40, 128)           0

 dropout (Dropout)           (None, 40, 128)           0

 max_pooling1d (MaxPooling1D  (None, 5, 128)           0
 )

 conv1d_2 (Conv1D)           (None, 5, 128)            82048

 activation_2 (Activation)   (None, 5, 128)            0

 conv1d_3 (Conv1D)           (None, 5, 128)            82048

 activation_3 (Activation)   (None, 5, 128)            0

 flatten (Flatten)           (None, 640)               0

 dense (Dense)               (None, 7)                 4487

 activation_4 (Activation)   (None, 7)                 0

=================================================================
Total params: 334,087
Trainable params: 334,087
Non-trainable params: 0
```

Prossimi step sono la compilazione del modello con "accuracy" come metrica di riferimento e il fit del modello con il supporto del testing e del training set. Come validation set è usato il testing set.

```
model.compile(loss='categorical_crossentropy', optimizer=opt,metrics=['accuracy'])
```

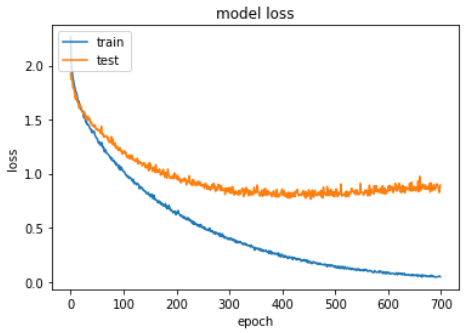
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)
```

```
cnnhistory=model.fit(X_train, y_train, batch_size=16, epochs=700, validation_data=(X_test, y_test))
```

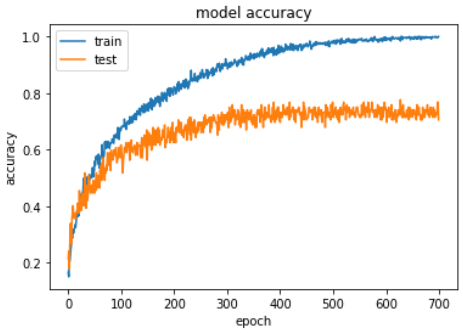
```
7232
Epoch 695/700
28/28 [==============================] - 1s 25ms/step - loss: 0.0466 - accuracy: 1.0000 - val_loss: 0.8997 - val_accuracy: 0.
7321
Epoch 696/700
28/28 [==============================] - 1s 36ms/step - loss: 0.0478 - accuracy: 0.9978 - val_loss: 0.8781 - val_accuracy: 0.
7232
Epoch 697/700
28/28 [==============================] - 1s 36ms/step - loss: 0.0502 - accuracy: 0.9955 - val_loss: 0.8464 - val_accuracy: 0.
7321
Epoch 698/700
28/28 [==============================] - 1s 36ms/step - loss: 0.0536 - accuracy: 0.9955 - val_loss: 0.8254 - val_accuracy: 0.
7679
Epoch 699/700
28/28 [==============================] - 1s 23ms/step - loss: 0.0476 - accuracy: 0.9978 - val_loss: 0.8561 - val_accuracy: 0.
7679
Epoch 700/700
28/28 [==============================] - 1s 20ms/step - loss: 0.0471 - accuracy: 1.0000 - val_loss: 0.8926 - val_accuracy: 0.
7054
```

A seguito del model.fit, è possibile osservare che nelle ultime epoche il valore di val_loss subisce molte oscillazioni, così come anche il valore assunto da val_accuracy. Le epoche sono state fissate ad un numero troppo elevato, sono state scelte 700 epoche, ma è possibile notare dai plot inseriti nella pagina seguente che, già intorno a 400 epoche, il modello aveva terminato l'apprendimento vero e proprio. Dall'epoca 400 in poi, il modello dimostra di riempirsi di dati, ma senza apprendere realmente. Nonostante questa considerazione, vediamo che i valori di val_loss decrescono e quelli di val_accuracy crescono, quindi in generale possiamo affermare che il modello porta avanti un buon apprendimento sui dati.

```
plt.plot(cnnhistory.history['loss'])
plt.plot(cnnhistory.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



```
plt.plot(cnnhistory.history['accuracy'])
plt.plot(cnnhistory.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```




```
model_name = 'Emotion_Voice_Detection_Model.h5'
save_dir = os.path.join(os.getcwd(), 'saved_models')
# Save model and weights
if not os.path.isdir(save_dir):
    os.makedirs(save_dir)
model_path = os.path.join(save_dir, model_name)
model.save(model_path)
print('Saved trained model at %s ' % model_path)
```

```
Saved trained model at C:\Users\fedes\Desktop\CODICE_TESI\saved_models\Emotion_Voice_Detection_Model.h5
```

```
import json
model_json = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)
```

```
# loading json and creating model
from keras.models import model_from_json
json_file = open('model.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)
# load weights into new model
loaded_model.load_weights("saved_models/Emotion_Voice_Detection_Model.h5")
print("Loaded model from disk")

# evaluate loaded model on test data
loaded_model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])
score = loaded_model.evaluate(X_test, y_test, verbose=0)
print("%s: %.2f%%" % (loaded_model.metrics_names[1], score[1]*100))
```

```
Loaded model from disk
accuracy: 70.54%
```

Il valore di accuracy generale del modello in esame è del 70.54%; un valore molto alto e soddisfacente per le analisi che si vogliono condurre.

```
preds = loaded_model.predict(X_test,
                         batch_size=32,
                         verbose=1)
```

```
4/4 [==============================] - 0s 5ms/step
```

```
preds
```

```
array([[...
        2.38717083e-04, 4.24019754e-06, 3.07321828e-02],
       [8.73849392e-01, 5.73119614e-03, 9.00554135e-02, 2.19753059e-03,
        2.53207926e-02, 3.09928524e-04, 2.53574853e-03],
       [1.47450134e-01, 1.21236779e-03, 5.98911755e-03, 2.66716368e-02,
        8.15828323e-01, 1.92740955e-03, 9.21015569e-04],
       [9.70759690e-02, 1.38518447e-03, 2.26786565e-02, 1.15024694e-03,
        1.25534916e-02, 8.15280139e-01, 4.98763472e-02],
       [1.78052171e-04, 6.92302883e-01, 1.32994101e-04, 4.27176292e-06,
        2.42123351e-01, 6.52584136e-02, 6.73803457e-10],
       [3.16084832e-01, 8.32215999e-04, 1.18483352e-02, 1.50804684e-01,
        4.95062384e-04, 2.20156810e-03, 5.17733335e-01],
       [2.67396681e-03, 1.48405170e-05, 9.88562942e-01, 2.06611585e-05,
        1.76683813e-03, 2.77534360e-03, 4.18545352e-03],
       [1.77783351e-02, 4.77691028e-05, 2.03449927e-05, 9.26738620e-01,
        2.10722942e-06, 5.50525263e-02, 3.60354577e-04],
       [7.15176109e-03, 9.61073150e-04, 8.24942708e-01, 1.73383788e-03,
        6.40209690e-02, 8.93336162e-02, 1.18560689e-02],
       [4.49658977e-03, 6.41672134e-01, 3.84028353e-05, 2.08111436e-04,
        8.83178873e-05, 3.53496373e-01, 1.15804774e-07]], dtype=float32)
```

```
preds1=preds.argmax(axis=1)
```

```
preds1
```

```
array([5, 1, 2, 5, 0, 3, 1, 0, 3, 2, 6, 5, 4, 5, 3, 6, 3, 3, 3, 5, 5, 1,
       6, 5, 0, 4, 3, 3, 3, 5, 4, 0, 2, 1, 1, 3, 4, 1, 4, 0, 2, 3, 1, 0,
       0, 6, 1, 5, 3, 1, 0, 6, 0, 4, 4, 6, 1, 2, 2, 4, 0, 3, 1, 4, 6, 0,
       0, 5, 4, 1, 5, 1, 2, 3, 6, 5, 1, 5, 3, 5, 2, 5, 4, 1, 3, 3, 5, 6,
       4, 4, 4, 0, 0, 3, 6, 3, 5, 1, 5, 6, 4, 2, 2, 0, 4, 5, 1, 6, 2, 3,
       2, 1], dtype=int64)
```

```
preds1=preds.argmax(axis=1)
```

```
preds1
```

```
array([5, 1, 2, 5, 0, 3, 1, 0, 3, 2, 6, 5, 4, 5, 3, 6, 3, 3, 3, 5, 5, 1,
       6, 5, 0, 4, 3, 3, 3, 5, 4, 0, 2, 1, 1, 3, 4, 1, 4, 0, 2, 3, 1, 0,
       0, 6, 1, 5, 3, 1, 0, 6, 0, 4, 4, 6, 1, 2, 2, 4, 0, 3, 1, 4, 6, 0,
       0, 5, 4, 1, 5, 1, 2, 3, 6, 5, 1, 5, 3, 5, 2, 5, 4, 1, 3, 3, 5, 6,
       4, 4, 4, 0, 0, 3, 6, 3, 5, 1, 5, 6, 4, 2, 2, 0, 4, 5, 1, 6, 2, 3,
       2, 1], dtype=int64)
```

```
preddf = pd.DataFrame({'predictedvalues': preds1})
preddf[:10]
```

	predictedvalues
0	5
1	1
2	2
3	5
4	0
5	3
6	1
7	0
8	3
9	2

```
actual=y_test.argmax(axis=1)
abc123 = actual.astype(int).flatten()
```

```
actualdf = pd.DataFrame({'actualvalues': abc123})
actualdf[:10]
```

	actualvalues
0	5
1	1
2	2
3	3
4	0
5	3
6	1
7	0
8	6
9	2

```
finaldf = actualdf.join(preddf)
```

```
finaldf.groupby('actualvalues').count()
```

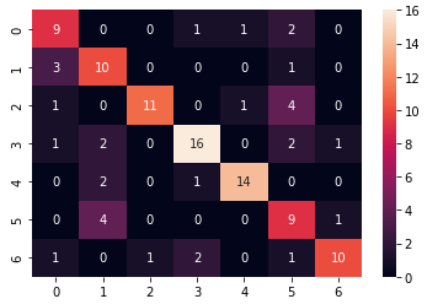
actualvalues	predictedvalues
0	13
1	14
2	17
3	22
4	17
5	14
6	15

```
finaldf.groupby('predictedvalues').count()
```

predictedvalues	actualvalues
0	15
1	18
2	12
3	20
4	16
5	19
6	12

Nella fase di predizione, sia dai valori nei dataframe visualizzati che dalla confusion matrix, ci accorgiamo che il match tra valori predetti e valori effettivi è molto alto. Tutte le classi sono ben predette, le uniche difficoltà che il modello mostra si vedono nell'errata interpretazione delle classi 5 e 2.

```
#visualizzazione matrix_confusion

cm = confusion_matrix(abc123,preds1)   #confronto valori label
f = sns.heatmap(cm, annot=True, fmt='d')
```




```
print(classification_report(abc123, preds1))
```

```
              precision    recall  f1-score   support

           0       0.60      0.69      0.64        13
           1       0.56      0.71      0.63        14
           2       0.92      0.65      0.76        17
           3       0.80      0.73      0.76        22
           4       0.88      0.82      0.85        17
           5       0.47      0.64      0.55        14
           6       0.83      0.67      0.74        15

    accuracy                           0.71       112
   macro avg       0.72      0.70      0.70       112
weighted avg       0.74      0.71      0.71       112
```

Infine, il classification report e la tabella di accuracy per classe sono inseriti a conferma di quanto già detto sopra. I valori di accuracy per classe minori sono quelli sulla classe 2 e sulla classe 5; ma sono valori che comunque rimangono al di sopra del 50%.

```
def classwise_accuracy():
        a = pd.crosstab(abc123,preds1)
        print(a.max(axis=1)/a.sum(axis=1))
classwise_accuracy()
```

```
row_0
0    0.692308
1    0.714286
2    0.647059
3    0.727273
4    0.823529
5    0.642857
6    0.666667
dtype: float64
```

4.4.1 Applicazione su lezioni in italiano del Politecnico

Come già detto, l'obiettivo ultimo è la predizione delle emozioni dei professori durante le loro spiegazioni. In questo paragrafo si rendono noti i risultati ottenuti dalle analisi effettuate su diversi corsi in italiano, ovvero quello di Elettrotecnica e quello di Sistemi.

```
import moviepy.editor
import random
import fnmatch
import sys
from moviepy.editor import VideoFileClip
from moviepy.editor import *
```

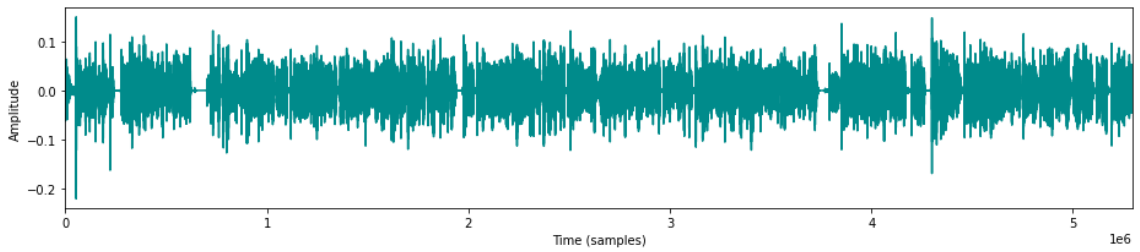
A partire dalle videolezioni sono state estratte randomicamente delle subclip con durata di 2 minuti. Inoltre è stata effettuata una conversione da mp4 a wav, questo perché librosa lavora con audio con estensione .wav.

```
directory = 'dataset'
ext = "*mp4"
output_ext='*wav'
xdim = 854
ydim = 480
length = 120

inputs = [os.path.join(directory,f) for f in os.listdir(directory) if os.path.isfile(os.path.join(directory, f))
          and fnmatch.fnmatch(f, ext)]


def subclip(video_file, output="mp4"):
    filename, ext = os.path.splitext(video_file)
    # import to moviepy
    clip = moviepy.editor.VideoFileClip(video_file).resize( (xdim, ydim) )

    # select a random time point
    start = round(random.uniform(0,clip.duration-length), 2)

    # cut a subclip
    out_clip = clip.subclip(start,start+length)

    out_clip.write_videofile(f"{filename}.{output}")

def mp4tomp3(mp4file, output_ext="wav"):
    filename, ext = os.path.splitext(mp4file)

    videoclip=VideoFileClip(mp4file)
    audioclip=videoclip.audio

    audioclip.write_audiofile(f"only_audio/{filename}.{output_ext}")
    audioclip.close()
    videoclip.close()


if __name__ == "__main__":
    for i in inputs:
        subclip(i)
        mp4tomp3(i)
```

Nel blocco di codice sottostante è stato inserito il plot di un audio come esempio.

```
%matplotlib inline



def print_plot_play(x, Fs, text=''):

    print('%s Fs = %d, x.shape = %s, x.dtype = %s' % (text, Fs, x.shape, x.dtype))
    plt.figure(figsize=(13, 3))
    plt.plot(x, color='darkcyan')
    plt.xlim([0, x.shape[0]])
    plt.xlabel('Time (samples)')
    plt.ylabel('Amplitude')
    plt.tight_layout()
    plt.show()
    ipd.display(ipd.Audio(data=x, rate=Fs))

# Read wav
fn_wav = os.path.join(f"only_audio/dataset\Elettrotecnica_ELT_lez_02.wav")
x, Fs = librosa.load(fn_wav, sr=None)
print_plot_play(x=x, Fs=Fs, text='WAV file: ')
```

```
WAV file:  Fs = 44100, x.shape = (5293323,), x.dtype = float32
```



Una volta effettuate le operazioni precedentemente descritte, si è passati all'estrazione delle features utili al fine dell'interpretazione emozionale. In particolare, sono state estratte un numero di n_mfcc=13 con un sample rate sr=22050*2.

```
directoryAudio='only_audio/dataset'

for filename in os.listdir(directoryAudio):
        f = os.path.join(directoryAudio, filename)
            # checking if it is a file
        if os.path.isfile(f):
            print(f)

            X, sample_rate = librosa.load(f, res_type='kaiser_fast', duration=0.46,sr=22050*2,offset=0.5)
            sample_rate = np.array(sample_rate)
            mfccs = np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=13),axis=0)
            featurelive = mfccs
            livedf2 = featurelive
            livedf2= pd.DataFrame(data=livedf2)
            livedf2 = livedf2.stack().to_frame().T
            print(livedf2)
            twodim= np.expand_dims(livedf2, axis=2)
            livepreds = loaded_model.predict(twodim,batch_size=32, verbose=1)
            print(livepreds)
            livepreds1=livepreds.argmax(axis=1)
            liveabc = livepreds1.astype(int).flatten()

            print(colored(liveabc, 'red'))
```

Facendo un esempio di quanto si ottiene grazie a questo blocco di codice, riportiamo sotto i dati derivati per due specifici audio; dati forniti al lettore come esempio.

```
only_audio/dataset\Elettrotecnica_ELT_lez_01.wav
          0         1         2         3         4         5   \
          0         0         0         0         0         0
0 -23.983807 -19.912058 -17.072556 -18.390871 -22.103371 -23.851086

          6         7         8         9   ...         30         31  \
          0         0         0         0  ...          0          0
0 -28.457737 -28.165081 -28.002598 -27.937265  ... -30.791317 -33.413456

          32         33         34         35         36         37  \
          0          0          0          0          0          0
0 -32.796432 -33.715408 -37.966984 -38.474766 -36.836033 -33.603825

          38         39
          0          0
0 -36.790318 -37.973869


[1 rows x 40 columns]
1/1 [==============================] - 0s 46ms/step
[[1.6737998e-07 3.6002049e-08 1.0857402e-03 1.6883038e-03 1.8705890e-04
  9.9701983e-01 1.8798768e-05]]
[5]

only_audio/dataset\Elettrotecnica_ELT_lez_02.wav
          0         1         2         3         4         5         6   \
          0         0         0         0         0         0         0
0 -42.536625 -42.784328 -43.801643 -43.28854 -43.662338 -45.203247 -46.266087

          7         8         9   ...         30         31         32  \
          0         0         0  ...          0          0          0
0 -46.519722 -45.699257 -43.869198  ... -40.334221 -36.818058 -30.906332

          33         34         35         36       37         38         39
          0          0          0          0        0          0          0
0 -26.756365 -25.993052 -25.723906 -25.418678 -23.7775 -22.714775 -22.296471

[1 rows x 40 columns]
1/1 [==============================] - 0s 41ms/step
[[4.4349149e-10 1.4665173e-12 1.1411203e-06 2.0106369e-05 3.6938301e-05
  9.9993145e-01 1.0425019e-05]]
[5]
```

Quanto mostrato sopra riguarda le lezioni di Elettrotecnica; per quanto riguarda le lezioni di Sistemi i dati ottenuti dalle predizioni sono i seguenti.

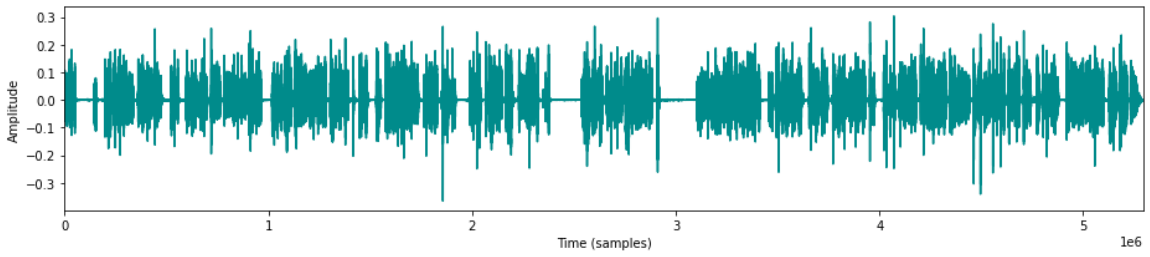
```
for filename in os.listdir(directoryAudio):
        f = os.path.join(directoryAudio, filename)
            # checking if it is a file
        if os.path.isfile(f):
            print(f)

            X, sample_rate = librosa.load(f, res_type='kaiser_fast', duration=0.46,sr=22050*2,offset=0.5)
            sample_rate = np.array(sample_rate)
            mfccs = np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=13),axis=0)
            featurelive = mfccs
            livedf2 = featurelive
            livedf2= pd.DataFrame(data=livedf2)
            livedf2 = livedf2.stack().to_frame().T
            print(livedf2)
            twodim= np.expand_dims(livedf2, axis=2)
            livepreds = loaded_model.predict(twodim,batch_size=32, verbose=1)
            print(livepreds)
            livepreds1=livepreds.argmax(axis=1)
            liveabc = livepreds1.astype(int).flatten()

            print(colored(liveabc, 'red'))
```

```
only_audio_sistemi/sistemi\Sistemi_elettronici_tecnologie_e_misure_lez_01.wav
          0          1          2          3          4          5   \
          0          0          0          0          0          0
0 -19.049854 -17.038288 -16.719887 -15.956355 -17.890511 -17.729431

          6          7          8          9   ...         30         31  \
          0          0          0          0  ...          0          0
0 -17.768126 -19.317347 -17.734907 -16.099195  ... -33.360954 -31.597979

          32         33         34         35         36         37  \
          0          0          0          0          0          0
0 -33.151043 -34.782589 -37.492741 -39.004704 -38.524742 -34.126122

          38         39
          0          0
0 -27.262308 -18.690004

[1 rows x 40 columns]
1/1 [==============================] - 0s 41ms/step
[[1.17380205e-05 6.47643958e-07 5.07532693e-02 1.96556859e-02
  6.03103545e-03 9.22995746e-01 5.51885401e-04]]
[5]
only_audio_sistemi/sistemi\Sistemi_elettronici_tecnologie_e_misure_lez_02.wav
          0         1          2          3         4          5          6   \
          0         0          0          0         0          0          0
0 -15.406937 -16.69417 -18.396473 -19.833635 -22.16968 -26.422268 -27.215191

          7          8          9   ...         30         31         32  \
          0          0          0  ...          0          0          0
0 -28.774496 -27.550865 -26.088011  ... -15.223539 -14.909304 -15.496885

          33         34         35         36         37         38         39
          0          0          0          0          0          0          0
0 -15.758142 -15.945543 -15.870668 -15.870703 -15.343029 -15.871716 -14.956023

[1 rows x 40 columns]
1/1 [==============================] - 0s 17ms/step
[[8.4710541e-07 1.0788289e-06 1.5418665e-04 6.8048248e-04 1.6191935e-03
  9.9687010e-01 6.7410764e-04]]
[5]
```

A supporto dei dati, anche in questo caso, inseriamo una rappresentazione di un audio derivato dalle videolezioni.

```
# Read wav
fn_wav = os.path.join(f"only_audio_sistemi/sistemi\Sistemi_elettronici_tecnologie_e_misure_lez_10.wav")
x, Fs = librosa.load(fn_wav, sr=None)
print_plot_play(x=x, Fs=Fs, text='WAV file: ')
```

```
WAV file:  Fs = 44100, x.shape = (5293323,), x.dtype = float32
```



4.4.2 Risultati ottenuti

Il seguente paragrafo mira ad argomentare i risultati trovati sopra e trovare delle conclusioni, che possano essere utilizzate anche in futuro per eventuali approfondimenti dell'argomento e per la ricerca di nuovi risvolti applicativi e/o teorici. Prima cosa che è possibile notare è che entrambi i modelli tirano fuori lo stesso risultato, ovvero tutte le lezioni di entrambi i corsi sono etichettate allo stesso modo, a loro è associata indistintamente la classe 5, ovvero è associata come emozione la sorpresa. In generale, trova una giustificazione questo risultato nel fatto che il professore in questione, durante le sue spiegazioni, utilizza un tono di voce squillante e molto alto. La sua voce è ampia e l'intensità è alta e mantenuta costante per tutta la lezione. Il modo di spiegare del professore risulta omogeneo, sia all'interno della stessa lezione che tra una lezione ed un'altra. Stessa cosa accade per le lezioni tenute dal professore di Sistemi, che mantiene un tono elevato per tutte le lezioni ed una voce ampia.

4.5 Applicazioni reti CNN e LSTM con vari dataset su audio tratti dalle lezioni svolte al Politecnico.

Dati i risultati non soddisfacenti derivati dai test con EMOVO su rete CNN, si è deciso di continuare le analisi, utilizzando registrazioni di lezioni in inglese. Considerazione che si vuole inserire è che il dataset EMOVO risulta molto debole per un riconoscimento emozionale sulle lezioni del Politecnico. EMOVO non è abbastanza robusto da riuscire a ricavare risultati efficienti. Nonostante il modello, alla fine dell'apprendimento, mostri un'accuracy generale di valore alto, parliamo di un risultato nell'intorno dell'80%, nel momento in cui viene effettuata l'operazione di predizione, esso non riesce a discernere tra le diverse possibili emozioni e non riesce a percepire se sono presenti o meno delle sfumature in quello che gli viene fornito come input. Sfumature che da un ascolto attento un umano riuscirebbe a cogliere; è importante precisare che quest'ultima frase verrà approfondita successivamente e

troverà a suo supporto una validazione svolta con soggetti umani scelti casualmente a cui sono state sottoposte le registrazioni. Tale validazione è presentata nel paragrafo 5.2.
Data la debolezza mostrata da EMOVO sulle predizioni, sono stati effettuati i test presentati a seguire.
Prima di tutto introduciamo i paragrafi dicendo che la rete utilizzata continua ad essere dello stesso tipo mostrato nel paragrafo 4.4, a cambiare sono i dataset utilizzati per l'apprendimento e i dati sui quali sono fatte le predizioni. Questa volta le lezioni su cui si vogliono produrre le predizioni sono le lezioni in inglese fornite dal Politecnico, in particolare i corsi analizzati sono: Advanced design for signal e Photonic devices.

4.5.1 Rete CNN con RAVDESS+TESS

Dopo la procedura di estrazioni di subclip dalle videolezioni di partenza e l'estrazione della parte audio da queste, operazioni che vengono condotte esattamente allo stesso modo di come sono state presentate nel paragrafo 4.4.1, è effettuata la predizione delle emozioni nel seguente modo:

```
directoryAudio='only_audio/inglese'

for filename in os.listdir(directoryAudio):
        f = os.path.join(directoryAudio, filename)
            # checking if it is a file
        if os.path.isfile(f):
            print(f)

            X, sample_rate = librosa.load(f, res_type='kaiser_fast', duration=0.46,sr=22050*2,offset=0.5)
            sample_rate = np.array(sample_rate)
            mfccs = np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=13),axis=0)
            featurelive = mfccs
            livedf2 = featurelive
            livedf2= pd.DataFrame(data=livedf2)
            livedf2 = livedf2.stack().to_frame().T
            print(livedf2)
            twodim= np.expand_dims(livedf2, axis=2)
            livepreds = loaded_model.predict(twodim,batch_size=32, verbose=1)
            print(livepreds)
            livepreds1=livepreds.argmax(axis=1)

            liveabc = livepreds1.astype(int).flatten()
            livepredictions = (lb.inverse_transform((liveabc)))
            print(livepredictions)
```

con risultati:

```
only_audio/inglese\Advanced_design_for_signal_lez_01.wav
          0         1         2         3         4         5         6   \
          0         0         0         0         0         0         0
0 -47.452106 -47.353741 -46.444515 -46.375488 -46.96299 -46.148994 -45.566677

          7         8         9   ...         30        31        32  \
          0         0         0  ...          0         0         0
0 -45.561623 -41.6768 -37.866161  ... -28.701054 -30.47706 -32.465385

          33        34        35        36        37        38        39
          0         0         0         0         0         0         0
0 -30.672464 -29.122728 -32.985588 -36.593239 -41.314861 -44.496635 -45.824295

[1 rows x 40 columns]
1/1 [==============================] - 0s 164ms/step
[[5.64825257e-07 1.85077333e-05 9.91931975e-01 8.03788751e-03
  1.09766224e-05 9.22096926e-13 9.05577609e-12 1.01427444e-22]]
['disgust']
```

```
only_audio/inglese\Advanced_design_for_signal_lez_02.wav
          0          1         2          3          4          5          6   \
          0          0         0          0          0          0          0
0 -41.227234 -41.708176 -40.44199 -40.587296 -41.496437 -45.315018 -43.398788

          7          8          9   ...         30         31       32  \
          0          0          0  ...          0          0        0
0 -41.956821 -41.167244 -40.218384  ... -45.470585 -44.135983 -46.4739

         33        34         35         36         37         38         39
          0         0          0          0          0          0          0
0 -46.37875 -45.01432 -43.907593 -44.080437 -44.869034 -46.434952 -45.652946

[1 rows x 40 columns]
1/1 [==============================] - 0s 38ms/step
[[5.0723628e-07 8.5639967e-06 9.9655020e-01 3.4387407e-03 2.0327507e-06
  3.2314411e-15 5.4831488e-15 2.4712197e-25]]
['disgust']
```

L'estrazione delle features dagli audio è di n_mfcc=13 ed il parametro di duration della librosa.feature.mfcc è pari a 0.46. Come accaduto per la rete CNN con dataset EMOVO e predizione su lezioni in italiano, notiamo che l'emozione estratta è sempre la stessa per tutti gli audio. Nel caso specifico, l'emozione predetta è di disgusto. L'interpretazione da parte del modello dell'emozione di disgusto come emozione del professore, trova conferma nella teoria presentata nella prima parte di questa tesi. Il disgusto è caratterizzato da scarsa variazione di tonalità e bassa velocità, ma soprattutto da forte segmentazione delle sillabe. Nonostante questa associazione teorica ed il fatto che il modello presenti un'accuracy generale del 54.17%, non riuscendo ad ottenere il risultato desiderato, passiamo ad analizzare un nuovo caso nel paragrafo successivo.

4.5.2 Rete CNN con RAVDESS+TESS e split=7

La rete continua ad essere quella CNN con le stesse caratteristiche e la stessa profondità, il dataset continua ad essere l'unione di Ravdess e Tess; la modifica che si è attuata è quella relativa allo splitting dei dati. Sono stati ricavati dal dataset di partenza il training set, il validation set e il testing set; il valore di n_fold preso in considerazione è 7. Tale scelta è stata dettata dai risultati presi per la rete in questione nel paragrafo 3.1.1. La rete ha un'accuracy generale del 34.77% e la predizione dell'emozione sulle videolezioni di Advanced design for signal

```
directoryAudio='only_audio/inglese'

for filename in os.listdir(directoryAudio):
        f = os.path.join(directoryAudio, filename)
            # checking if it is a file
        if os.path.isfile(f):
            print(f)

            X, sample_rate = librosa.load(f, res_type='kaiser_fast', duration=0.46,sr=22050*2,offset=0.5)
            sample_rate = np.array(sample_rate)
            mfccs = np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=13),axis=0)
            featurelive = mfccs
            livedf2 = featurelive
            livedf2= pd.DataFrame(data=livedf2)
            livedf2 = livedf2.stack().to_frame().T
            print(livedf2)
            twodim= np.expand_dims(livedf2, axis=2)
            livepreds = loaded_model.predict(twodim,batch_size=32, verbose=1)
            print(livepreds)
            livepreds1=livepreds.argmax(axis=1)

            liveabc = livepreds1.astype(int).flatten()
            livepredictions = (lb.inverse_transform((liveabc)))
            print(livepredictions)
```

offre i seguenti risultati:

```
only_audio/inglese\Advanced_design_for_signal_lez_01.wav
          0          1          2          3          4          5          6   \
          0          0          0          0          0          0          0
0 -47.452106 -47.353741 -46.444515 -46.375488 -46.96299 -46.148994 -45.566677

          7          8          9   ...         30         31         32  \
          0          0          0  ...          0          0          0
0 -45.561623 -41.6768 -37.866161  ... -28.701054 -30.47706 -32.465385

          33         34         35         36         37         38         39
          0          0          0          0          0          0          0
0 -30.672464 -29.122728 -32.985588 -36.593239 -41.314861 -44.496635 -45.824295

[1 rows x 40 columns]
1/1 [==============================] - 0s 71ms/step
[[1.7335893e-01 8.0021888e-01 2.1728475e-03 4.1897870e-03 1.7905630e-02
  2.4672697e-04 1.8237367e-03 8.3522515e-05]]
['calm']

only_audio/inglese\Advanced_design_for_signal_lez_02.wav
          0          1          2          3          4          5          6   \
          0          0          0          0          0          0          0
0 -41.227234 -41.708176 -40.44199 -40.587296 -41.496437 -45.315018 -43.398788

          7          8          9   ...         30         31       32  \
          0          0          0  ...          0          0        0
0 -41.956821 -41.167244 -40.218384  ... -45.470585 -44.135983 -46.4739

         33         34         35         36         37         38         39
          0          0          0          0          0          0          0
0 -46.37875 -45.01432 -43.907593 -44.080437 -44.869034 -46.434952 -45.652946

[1 rows x 40 columns]
1/1 [==============================] - 0s 15ms/step
[[9.9056616e-02 8.9410627e-01 7.1241241e-04 1.2911082e-03 4.4849240e-03
  6.2861211e-05 2.6508069e-04 2.0672926e-05]]
['calm']
```

L'estrazione delle features dagli audio è di n_mfcc=13 e il parametro di duration della librosa.feature.mfcc è pari a 0.46. Deduciamo dai dati che per tutte le lezioni il modello di apprendimento automatico tira fuori sempre la stessa emozione, ovvero la calma. Anche in questo caso, è possibile giustificare quanto tirato fuori dalla rete, perchè la calma è un'emozione che potrebbe essere associata al professore durante la sua spiegazione, essendo molto lento nella presentazione degli argomenti, facendo molte pause e tenendo la spiegazione sempre su un tono di voce basso. Ancora una volta si vuole precisare che la spiegazione dei risultati verrà approfondita e ripresa nel Capitolo 5.

4.5.3 Rete CNN con RAVDESS+SAVEE

Dato che i risultati ottenuti nel paragrafo 4.5.2 e 4.5.1 non ci soddisfano si è deciso di effettuare le analisi utilizzando un dataset diverso, ovvero quello ottenuto dall'unione di Ravdess e Savee. Sempre con l'utilizzo della stessa rete, la predizione questa volta ci permette di avere più risultati da analizzare e ci permette di fare considerazioni diverse. La rete in questione, adesso, presenta un'accuracy generale del 38.22% e permette di avere dei risultati molto più interessanti da discutere.

I risultati della predizione sono quelli inseriti di seguito:

```
only_audio/inglese\Advanced_design_for_signal_lez_01.wav
          0         1         2         3         4         5         6    \
          0         0         0         0         0         0         0
0 -47.396828 -47.29398 -46.385597 -46.331356 -46.908508 -46.087944 -45.505692

          7         8         9    ...       206       207       208  \
          0         0         0  ...         0         0         0
0 -45.502537 -41.626118 -37.807823  ... -46.25927 -47.846581 -49.601955

          209       210       211       212       213       214       215
          0         0         0         0         0         0         0
0 -48.923153 -47.355587 -48.518291 -47.545097 -47.665493 -47.830986 -48.703621

[1 rows x 216 columns]
1/1 [==============================] - 0s 18ms/step
[[2.55934452e-03 2.36597839e-06 1.71170577e-01 6.60216331e-01
  1.18581556e-01 3.06676156e-05 6.97591034e-08 1.26062371e-02
  1.49396695e-02 1.98931992e-02]]
['female_happy']
only_audio/inglese\Advanced_design_for_signal_lez_02.wav
          0         1         2         3         4         5    \
          0         0         0         0         0         0
0 -40.610512 -40.940388 -39.657974 -39.781155 -40.672691 -44.533695

          6         7         8         9    ...       206       207  \
          0         0         0         0  ...         0         0
0 -42.613132 -41.138481 -40.373642 -39.432217  ... -45.290485 -43.119347

          208       209       210       211       212       213  \
          0         0         0         0         0         0
0 -43.106621 -43.171635 -41.896324 -42.560417 -42.667107 -42.913475

          214       215
          0         0
0 -43.811905 -43.393513


[1 rows x 216 columns]
1/1 [==============================] - 0s 17ms/step
[[1.1163731e-02 8.4635591e-05 1.2992065e-02 5.3950155e-01 2.5645110e-01
  1.2569921e-03 2.7422900e-06 8.8043455e-03 1.6185001e-01 7.8928862e-03]]
['female_happy']
only_audio/inglese\Advanced_design_for_signal_lez_03.wav
          0         1         2         3         4         5         6    \
          0         0         0         0         0         0         0
0 -24.952299 -26.518497 -27.543736 -27.09568 -28.101204 -26.932228 -28.793285

          7         8         9    ...       206       207       208  \
          0         0         0  ...         0         0         0
0 -30.669279 -32.738605 -32.478062  ... -42.817257 -42.609566 -42.456459

          209       210       211       212       213       214       215
          0         0         0         0         0         0         0
0 -43.497677 -44.505108 -43.277927 -44.258965 -43.331333 -44.560787 -47.373093

[1 rows x 216 columns]
1/1 [==============================] - 0s 17ms/step
[[2.5186488e-01 1.6902546e-04 9.6760485e-03 9.2288647e-03 5.7688636e-01
  7.1169809e-05 2.1697967e-07 7.8037351e-02 3.2793142e-02 4.1273035e-02]]
['female_sad']
only_audio/inglese\Advanced_design_for_signal_lez_04.wav
          0         1         2         3         4         5    \
          0         0         0         0         0         0
0 -33.400986 -33.954868 -31.148603 -27.484081 -27.795319 -29.785343

          6         7         8         9    ...       206       207  \
          0         0         0         0  ...         0         0
0 -28.695818 -30.173933 -33.603268 -34.104233  ... -43.234673 -43.399826

          208       209       210       211       212       213       214  \
          0         0         0         0         0         0         0
0 -42.169651 -41.829594 -41.12524 -41.708908 -42.800041 -44.709778 -43.336491

          215
          0
0 -42.957706

[1 rows x 216 columns]
1/1 [==============================] - 0s 16ms/step
[[3.9138544e-02 1.3098229e-05 7.1837193e-01 1.7062698e-01 1.8835017e-02
  2.7497701e-04 6.9259059e-08 4.2115297e-02 6.5503200e-03 4.0736301e-03]]
['female_fearful']
```

```
only_audio/inglese\Advanced_design_for_signal_lez_05.wav
          0          1          2          3          4          5          6    \
            0          0          0          0          0          0          0
0 -46.817348 -46.306984 -46.791283 -46.204002 -42.84391 -43.636902 -44.780075

          7          8          9    ...        206        207        208  \
            0          0          0  ...          0          0          0
0 -43.688168 -38.132366 -29.938541  ... -45.285942 -44.038277 -43.844994

          209        210        211        212        213        214        215
            0          0          0          0          0          0          0
0 -44.629513 -45.171993 -45.22398 -44.972569 -43.8489 -43.351891 -43.892803

[1 rows x 216 columns]
1/1 [==============================] - 0s 17ms/step
[[6.3236796e-05 1.1979390e-06 3.1484936e-03 6.0660828e-02 9.3484633e-02
  1.5102417e-05 9.1151593e-11 6.0664499e-03 8.2106078e-01 1.5499262e-02]]
['male happy']
only_audio/inglese\Advanced_design_for_signal_lez_06.wav
          0          1          2          3          4          5          6    \
            0          0          0          0          0          0          0
0 -26.517872 -29.093956 -32.61113 -31.421877 -35.43993 -37.668091 -36.495571

          7          8          9    ...        206        207        208  \
            0          0          0  ...          0          0          0
0 -36.698982 -33.530754 -29.418335  ... -33.409355 -34.539516 -33.088638

          209        210        211        212        213        214        215
            0          0          0          0          0          0          0
0 -32.719765 -30.56362 -27.731947 -26.407104 -24.677174 -24.406477 -22.474459

[1 rows x 216 columns]
1/1 [==============================] - 0s 19ms/step
[[4.3667260e-05 2.2805743e-06 3.1175715e-04 8.7479985e-04 2.6609682e-02
  3.6431542e-05 4.0883479e-06 5.4016425e-03 3.9269611e-01 5.7401949e-01]]
['male_sad']
only_audio/inglese\Advanced_design_for_signal_lez_07.wav
          0          1          2          3          4          5    \
            0          0          0          0          0          0
0 -42.217693 -40.648972 -40.147068 -40.283638 -41.099529 -42.374283

          6          7          8          9    ...        206        207  \
            0          0          0          0  ...          0          0
0 -39.941452 -39.74778 -41.659458 -40.114765  ... -44.056839 -43.214195

          208        209        210        211        212        213  \
            0          0          0          0          0          0
0 -42.562691 -42.120178 -43.311092 -42.406029 -40.774658 -39.669426

          214        215
            0          0
0 -41.049614 -43.295876

[1 rows x 216 columns]
1/1 [==============================] - 0s 17ms/step
[[3.6239397e-02 8.5471320e-06 5.1122165e-04 2.8800955e-06 2.1253046e-04
  1.4588677e-03 3.1827319e-06 9.4654572e-01 7.4901240e-04 1.4268607e-02]]
['male_fearful']
```

La conferma che l'attendibilità del modello in questione non sia pari o superiore al 50% ci arriva dai dati predetti, infatti per metà delle lezioni il sesso riconosciuto è quello femminile, quando in realtà le lezioni sono condotte tutte da un professore, e quindi, da una persona di sesso maschile. Per quanto riguarda l'aspetto emotivo, vengono predette felicità, tristezza, paura e, poi, di nuovo felicità, tristezza e paura. Le emozioni sono giustificabili dal punto di vista teorico in quanto quando il tono di voce si fa più forte ed il professore è più sicuro anche di quello che sta dicendo e non mostra dubbi viene predetta la felicità. Al contrario, quando ci sono incertezze in quello che sta dicendo, il tono si fa più basso, la velocità del suo parlato diminuisce e c'è un balbettio che caratterizza le sue frasi in modo frequente. Più volte il professore ripete la stessa parola, oppure torna indietro e riprende la frase per continuarla; tutta questa sua incertezza viene interpretata dalla rete di apprendimento come paura oppure tristezza. La voce è spezzata e stretta, la segmentazione delle sillabe è molto marcata, l'intensità bassa e tutto questo ha come conseguenza una predizione del tipo detto sopra.

In più, l'incertezza del professore è testimoniata dai numerosissimi intercalari che utilizza; per la maggior parte riscontriamo, appunto, intercalari legati al dubbio come "ehm, mmh, ee..". Ovviamente non ci rifacciamo al senso delle parole usate, perchè il nostro studio prescinde da questo, però l'intonazione e la sillabazione di tali intercalari è fondamentale per il riconoscimento di un'emozione, quale quella della paura.

Sulla stessa rete è stato effettuato, anche, il riconoscimento delle emozioni per le videolezioni del corso Photonic devices ed i risultati ottenuti sono dei migliori, in quanto il modello riesce a interpretare correttamente per ogni caso il giusto sesso dell'interlocutore, ovvero maschile. Altro aspetto importante è che il modello interpreta felicità, paura e per la maggior parte dei casi tristezza. L'interpretazione delle emozioni sembra corretta, infatti la gioia è riscontrata quando il tono di voce è più elevato e la voce più squillante, con scansione delle sillabe maggiore; mentre la tristezza e la paura sono associate ad un tono di voce più basso e ad un ritmo più lento nel pronunciare le parole. Stessa interpretazione è stata data dagli agenti umani coinvolti nella validazione dei risultati, che verrà mostrata tra qualche paragrafo in questa trattazione.

4.5.4 Rete CNN con RAVDESS+SAVEE e split=7

Ultimo test svolto è quello con stessa rete e stesso dataset di riferimento del sottoparagrafo 4.5.3, unica differenza è lo splitting del set di dati in n_fold=7. Con l'introduzione di questa modifica, i dati ottenuti sono del tipo:

```
only_audio/inglese\Advanced_design_for_signal_lez_01.wav
          0         1         2         3         4         5         6    \
            0         0         0         0         0         0         0
0 -47.396828 -47.29398 -46.385597 -46.331356 -46.908508 -46.087944 -45.505692

          7         8         9    ...       206       207       208  \
            0         0         0  ...         0         0         0
0 -45.502537 -41.626118 -37.807823  ... -46.25927 -47.846581 -49.601955

          209       210       211       212       213       214       215
            0         0         0         0         0         0         0
0 -48.923153 -47.355587 -48.518291 -47.545097 -47.665493 -47.830986 -48.703621

[1 rows x 216 columns]
1/1 [==============================] - 0s 205ms/step
[[0.02639183 0.00542863 0.4975787  0.21643159 0.07507832 0.00110332
  0.00106325 0.09387547 0.04373129 0.03931754]]
['female fearful']


only_audio/inglese\Advanced_design_for_signal_lez_02.wav
          0         1         2         3         4         5    \
            0         0         0         0         0         0
0 -40.610512 -40.940388 -39.657974 -39.781155 -40.672691 -44.533695

          6         7         8         9    ...       206       207  \
            0         0         0         0  ...         0         0
0 -42.613132 -41.138481 -40.373642 -39.432217  ... -45.290485 -43.119347

          208       209       210       211       212       213  \
            0         0         0         0         0         0
0 -43.106621 -43.171635 -41.896324 -42.560417 -42.667107 -42.913475

          214       215
            0         0
0 -43.811905 -43.393513
.                   .
1/1 [==============================] - 0s 71ms/step
[[0.05614736 0.01930905 0.24756911 0.1637926  0.20073931 0.00230047
  0.00690213 0.06905743 0.14120162 0.09298099]]
['female_fearful']
```

```
only_audio/inglese\Advanced_design_for_signal_lez_03.wav
          0          1          2          3          4          5          6    \
            0          0          0          0          0          0          0
0 -24.952299 -26.518497 -27.543736 -27.09568 -28.101204 -26.932228 -28.793285

          7          8          9    ...        206        207        208  \
            0          0          0  ...          0          0          0
0 -30.669279 -32.738605 -32.478062  ... -42.817257 -42.609566 -42.456459

          209        210        211        212        213        214        215
            0          0          0          0          0          0          0
0 -43.497677 -44.505108 -43.277927 -44.258965 -43.331333 -44.560787 -47.373093

[1 rows x 216 columns]
1/1 [==============================] - 0s 49ms/step
[[3.6424298e-02 3.2442577e-02 1.9568530e-01 1.4041656e-01 5.7510751e-01
  3.3392353e-04 5.5794086e-04 5.5863885e-03 5.1221899e-03 8.3233928e-03]]
['female_sad']
```

```
only_audio/inglese\Advanced_design_for_signal_lez_04.wav
          0          1          2          3          4          5    \
            0          0          0          0          0          0
0 -33.400986 -33.954868 -31.148603 -27.484081 -27.795319 -29.785343

          6          7          8          9    ...        206        207  \
            0          0          0          0  ...          0          0
0 -28.695818 -30.173933 -33.603268 -34.104233  ... -43.234673 -43.399826

          208        209        210        211        212        213        214  \
            0          0          0          0          0          0          0
0 -42.169651 -41.829594 -41.12524 -41.708908 -42.800041 -44.709778 -43.336491

          215
            0
0 -42.957706

1/1 [==============================] - 0s 47ms/step
[[0.07930356 0.01982674 0.35242116 0.18025498 0.28645682 0.00323239
  0.00167447 0.02095073 0.02272048 0.03315864]]
['female_fearful']
```

```
only_audio/inglese\Advanced_design_for_signal_lez_05.wav
          0          1          2          3          4          5          6    \
            0          0          0          0          0          0          0
0 -46.817348 -46.306984 -46.791283 -46.204002 -42.84391 -43.636902 -44.780075

          7          8          9    ...        206        207        208  \
            0          0          0  ...          0          0          0
0 -43.688168 -38.132366 -29.938541  ... -45.285942 -44.038277 -43.844994

          209        210        211        212        213        214        215
            0          0          0          0          0          0          0
0 -44.629513 -45.171993 -45.22398 -44.972569 -43.8489 -43.351891 -43.892803

[1 rows x 216 columns]
1/1 [==============================] - 0s 43ms/step
[[1.0743752e-02 1.5030109e-02 3.3014357e-01 3.8665092e-01 1.7054312e-01
  2.5910491e-04 1.3522451e-03 2.5596315e-02 2.3470217e-02 3.6210708e-02]]
['female_happy']
```

```
only_audio/inglese\Advanced_design_for_signal_lez_06.wav
          0          1          2          3          4          5          6    \
            0          0          0          0          0          0          0
0 -26.517872 -29.093956 -32.61113 -31.421877 -35.43993 -37.668091 -36.495571

          7          8          9    ...        206        207        208  \
            0          0          0  ...          0          0          0
0 -36.698982 -33.530754 -29.418335  ... -33.409355 -34.539516 -33.088638

          209        210        211        212        213        214        215
            0          0          0          0          0          0          0
0 -32.719765 -30.56362 -27.731947 -26.407104 -24.677174 -24.406477 -22.474459

[1 rows x 216 columns]
1/1 [==============================] - 0s 50ms/step
[[2.4519391e-02 3.7590008e-02 5.2881157e-01 1.5851606e-01 2.1749276e-01
  2.9796033e-04 2.9494527e-03 6.7107440e-03 7.7914582e-03 1.5320550e-02]]
['female_fearful']
```

```
only_audio/inglese\Advanced_design_for_signal_lez_07.wav
          0          1          2          3          4          5    \
            0          0          0          0          0          0
0 -42.217693 -40.648972 -40.147068 -40.283638 -41.099529 -42.374283

          6          7          8          9    ...        206        207  \
            0          0          0          0  ...          0          0
0 -39.941452 -39.74778 -41.659458 -40.114765  ... -44.056839 -43.214195

          208        209        210        211        212        213  \
            0          0          0          0          0          0
0 -42.562691 -42.120178 -43.311092 -42.406029 -40.774658 -39.669426

          214        215
            0          0
0 -41.049614 -43.295876

[1 rows x 216 columns]
1/1 [==============================] - 0s 57ms/step
[[0.09009552 0.02885424 0.3142544  0.10761594 0.12443499 0.00729998
  0.01195807 0.10722236 0.04489243 0.16337202]]
['female fearful']
```

L'estrazione delle features avviene su un audio di durata di 2.5 e con un numero n_mfcc=13. L'accuracy del modello generale è del 33.61%, a questo valore di accuracy, inferiore rispetto al modello presentato nel sottoparagrafo precedente, il modello reagisce effettuando una predizione del sesso della persona errata in tutti i casi. È predetto il sesso dell'interlocutore come femminile, quando sappiamo che il sesso è maschile. Tuttavia, è interessante osservare che le emozioni predette sono le stesse del modello precedente con cui si sta effettuando il confronto. Le emozioni predette, ancora una volta, sono tristezza, paura e gioia. Questo ci permette di dire che quel determinato professore, nelle sue lezioni, è molto probabile che abbia davvero un'emozione di quel tipo.

Si passa, adesso, a discutere i risultati trovati nei test di questo capitolo, per capire che conclusioni è possibile trarre ed attuarne dei confronti. Il Capitolo 5 vuole avere come obiettivo quello di tirare le somme di quanto mostrato in tutta la trattazione e portare a delle conclusioni sul lavoro svolto e sull'obiettivo raggiunto.

Conclusione

Il presente capitolo rappresenta il capitolo conclusivo della tesi scritta; in questo capitolo gli argomenti trattati, in ordine di inserimento, sono i seguenti:

- approccio metodologico e impostazione delle metriche fondamentali.
- presentazione dei risultati ottenuti, riassumendo la parte delle predizioni ottenute dagli audio estratti dalle lezioni dei corsi del Politecnico.
- considerazioni finali su quanto desunto dai test.
- validazione dei risultati ottenuti, in modo da comprendere quanto di quello predetto da un modello automatico di apprendimento come quello realizzato sia paragonabile e confrontabile con valutazioni umane.
- conclusioni e svolgimenti futuri.

5.1 Approccio metodologico, Applicazioni e Considerazioni finali

Nel presente paragrafo tracciamo le linee guida di quello che è stato l'argomento di studio affrontato e quali approcci metodologici sono stati utilizzati al fine di raggiungere l'obiettivo che ci si era prefissati, ovvero l'applicazione di un modello di apprendimento automatico sull'audio di videolezioni del Politecnico di Torino. Inoltre, si farà un breve excursus sulle metriche che si è dedotto giochino un ruolo fondamentale ai fini dell'apprendimento di una rete neurale per la Speech Emotion Recognition e saranno inserite delle considerazioni finali su quanto dedotto dagli esperimenti.

L'analisi degli algoritmi per la risoluzione del problema della Speech Emotion Recognition ci ha portato, in primo luogo, a fare una discriminazione tra reti di tipo CNN e reti di tipo LSTM. A seguito dei test condotti su dataset differenti e con modifiche differenti, apportate sia sulla costituzione della rete neurale che sui dati passati in input durante l'apprendimento al modello, si è prediletti le reti di tipo CNN senza operazioni di splitting sui dati o con valori di splitting non troppo elevati. Si è osservato che i valori di splitting generali, compatibili con qualsiasi dataset, sono quelli tra 10 e 20; ma i dataset con cui si è scelto di lavorare, avendo dimensioni non troppo vaste, permettevano miglioramenti in fase di apprendimento con valori di n_fold inferiori a 7. In alcuni casi, per dataset più ampi, si è visto che miglioramenti erano riscontrabili anche con valori di n_fold uguali a 10 e/o 15 e/o 20. Si è, poi, cercato di vedere se modificando la complessità della rete, aumentandone il numero di livelli, si potessero raggiungere valori di accuratezza dei modelli superiori, ma si è visto che aumentando i livelli si otteneva solo una complessità maggiore della rete. Quest'ultimo concetto si spiega con il fatto che, aumentando la complessità di una rete, non migliorano le prestazioni perchè si deve operare sempre entro certi limiti. Oltre un determinato limite, che è quello sopportato dalla rete, si va incontro ad un sovradattamento dei dati; avere più funzionalità estratte permette di risolvere problemi più complessi, ma introduce anche irregolarità nei dati.

Stesso comportamento si registra con un aumento delle epoche, è vero che un numero di epoche maggiore può comportare miglioramenti, ma è anche vero che stoppare prima un training può non far vedere eventuali miglioramenti che potevano verificarsi epoche dopo lo stop. È anche vero che un numero troppo elevato di epoche può portare al verificarsi di situazioni come l'overfitting. Il modello in questi casi si riempie di dati, come è capitato in molti test effettuati; ma non apprende nulla. In generale, l'obiettivo che ci si prefissa al termine del fit di un modello è la convergenza, quando questa si verifica o quando la tendenza della val_loss e della val_accuracy sono asintoticamente corrette, allora è quello l'intervallo di training in cui l'apprendimento sta dando ottimi risultati. Una volta considerate tutte le metriche fondamentali, è stato realizzato il modello con rete CNN per l'applicazione effettiva ai fini della tesi. Il primo obiettivo raggiunto con questa trattazione è stato quello di identificare i parametri e le metriche e, in generale, le modifiche apportabili ad un modello al fine di creare una rete che ci permettesse di risolvere il problema del riconoscimento emotivo a partire da audio nel migliore dei modi. Secondo obiettivo raggiunto è rappresentato dall'applicazione del modello realizzato sulle lezioni del Politecnico. Al termine di tale applicazione, i risultati ottenuti sono stati diversi, tutti hanno trovato un fondamento nella teoria e nelle considerazioni fatte nella prima parte della tesi, ovvero quella contenente il background teorico. Tutti i test applicativi sono stati condotti su una rete di tipo CNN con dataset differenti, EMOVO per le lezioni in italiano e Ravdess+Tess e Ravdess+Savee per le lezioni in inglese.

Tracciato il filo logico fondamentale per concludere quanto trovato nella tesi, passiamo a capire cosa è stato predetto dai test del capitolo 4 e perchè. Prima considerazione che va fatta, che ci fa capire perchè i risultati ottenuti dai test ci creino non poche incertezze, è quella relativa alle videolezioni in sé e per sé. Le registrazioni dei professori, che vengono utilizzate in ambito applicativo, introducono problematiche non indifferenti sulle predizioni. Prima di tutto sono registrazioni ricche di rumore, a causa dell'ambiente in cui sono effettuate. Si tratta di registrazioni fatte in aula, con rumori di sottofondo ed echi creati dall'ambiente, che non possono non essere considerati. L'interpretazione dei dati è influenzata dalla presenza di echi, in quanto gli echi mostrano picchi prominenti nello spettro di ricorrenza nell'analisi di segnali temporali. Oltre ai rumori e agli echi, altra problematica è quella introdotta dai microfoni con cui vengono effettuate le registrazioni, che non isolano così bene la voce dell'interlocutore. Ultimo, ma non meno importante, è il problema derivato dal fatto che durante la registrazione i professori tendono a muoversi, ad avvicinare o allontanare il microfono. Dopo aver detto ciò, sicuramente ci viene facile dedurre che, se le registrazioni dei dataset su cui fittiamo il modello sono dataset "puliti", quelli su cui facciamo la predizione non lo sono affatto. Altra questione è che i dataset su cui svolgiamo l'apprendimento sono interpretati da attori, che tendono ad enfatizzare le emozioni; mentre nel quotidiano qualsiasi interlocutore non potrà mai raggiungere quella stessa enfasi, se non in rari casi ed in determinati momenti. Questo ci porta a dire che, a meno di qualche momento in cui realmente il tono, l'intensità, la voce del professore cambiano per uno specifico motivo; per il resto rimane molto omogeneo il suo parlare così come rimane

coerente da una lezione ad un'altra anche il suo modo di spiegare. Così spieghiamo perchè da uno stesso professore traiamo fuori le stesse emozioni o poche sfumature emozionali. Altro aspetto findamentale è costituito dal fatto che, nelle lezioni ci sono tante parti che contengono pause, oppure intercalari, che non possono essere ben interpretate dal modello e che determinano ambiguità nella predizione. Inoltrandoci più nel tecnico, nel momento in cui deve essere svolta la predizione sull'audio isolato, parametri importanti sono quelli all'interno della funzione librosa.feature.mfcc, in particolare ci soffermiamo sulla duration, su n_mfcc e sul batch size. Si nota che quando la durata è maggiore, quindi quando viene presa in considerazione una porzione più ampia dell'audio, le predizioni sono migliori rispetto a quando la porzione di audio considerata è minore. Questo accade perché il modello ha bisogno di un maggiore intervallo per discernere un'emozione da un'altra, essendo audio molto rumorosi, ricchi di pause e di intercalari con cui il modello non riesce ad interfacciarsi. Arrivati a questo punto, visto che sono stati citati, attuiamo una piccola digressione in materia di intercalari e del modo di esprimersi di un interlocutore qualsiasi. Gli intercalari, che l'uomo tende ad usare frequentemente nel parlare quotidiano, vengono alle volte interpretate dal modello come disgusto, questo perchè intercalari come "ehm, ee" vengono confusi con suoni di disgusto, basti pensare all'assonanza che potrebbe esserci con intercalari come "bleah"; altre volte vengono accostati alla paura per via dell'incertezza che la persona manifesta. Stessa cosa vale per le ripetizioni oppure le parole spezzate e ripetute più volte, nel caso delle lezioni di Elettrotecnica, quando il professore è incerto sulle parole in inglese da usare e tende a balbettare, l'interpretazione fornita dal modello è quella di paura e timore. Terminata questa digressione, che sembrava doveroso fare, ricordiamo che ad influenzarci nella scelta dei valori e parametri a livello di predizione, sono i valori e i parametri fissati nella rete nel momento della sua creazione e del suo fit. Essendo influenzati da ciò, in alcuni test la duration era di 0.46 e in altri era di 2.5.

5.2 Validazione umana dei risultati ottenuti e Confronto con dati predetti dal modello di apprendimento automatico

Per validare e comprendere quanto siano validi e conformi alla realtà i risultati predetti dai modelli presentati ed illustrati nel Capitolo 4, si è deciso di confrontare tali predizioni con i dati raccolti da un test condotto su agenti umani.

I risultati del test condotto sono consultabili al link https://www.dropbox.com/sh/lzjgxg5g5uzuu7a/AABMGJLXuP3afMZRq7a9ZoI8a?dl=0

È stato chiesto a 15 ragazzi di ascoltare gli audio estratti dalle videolezioni, su cui sono state fatte le predizioni, e di definire che tipo di emozione a loro parere era espressa dall'interlocutore. A ciascuno di loro sono state date come possibilità di scelta le emozioni che riescono ad interpretare i modelli analizzati in questa tesi, con l'aggiunta delle opzioni "nessuna emozione tra quelle indicate" e la possibilità di inserire un'emozione, qualora non

fosse presente tra quelle in lista. Sono state poste ai soggetti, anche, domande varie per capire le motivazioni delle loro scelte e sono state ascoltate eventuali considerazioni da loro fatte sugli ascolti.

In generale, tutti i soggetti interrogati hanno espresso il dubbio di presenza di una specifica emozione, affermando che da quelle videolezioni non ritenevano estraibili stati emotivi. Nonostante questo loro primo approccio, ponendoli di fronte ad una scelta obbligata, riuscivano, riascoltando attentamente, ad identificare una o più emozioni per ciascun audio.
Nel caso degli audio di Elettrotecnica, l'emozione maggiormente associata era quella di calma, tutti i soggetti hanno chiarito che questa loro scelta era dovuta al tono costante utilizzato dal professore durante la spiegazione ed alla formulazione di frasi senza variazioni di velocità nel parlare. Altra emozione associata era quella di gioia o di sorpresa, in quanto il tono del docente risultava alto e l'intensità, allo stesso modo, elevata. Dall'ascolto delle lezioni, tutti hanno escluso fortemente emozioni come la rabbia o la tristezza. Nelle valutazioni c'è stato, invece, un alternarsi nella percezione della calma e della gioia; questo perchè nei momenti di spiegazione dei calcoli alla lavagna la maggior parte dei soggetti percepiva solamente la calma della spiegazione, mentre nei momenti di dialogo diretto con gli studenti il tono usato per comunicare veniva associato ad emozioni più forti come gioia e sorpresa. Alcuni hanno aggiunto noia tra le emozioni possibili, rilevata durante la spiegazione delle formule e quando il professore interfacciandosi con l'allievo chiedeva conferma nella comprensione di quanto da lui spiegato. In linea generale, tutti i soggetti a domanda posta, hanno risposto che il mood della spiegazione del professore era da ritenersi omogeneo sia all'interno della stessa lezione, che tra lezioni diverse.

Nel caso delle registrazioni di Sistemi elettronici: tecnologie e misure, la maggior parte dei ragazzi consultati ha distinto la tristezza come emozione comune negli audio, dovuta ad un tono di voce basso utilizzato durante tutta la spiegazione, altra emozione riscontrata è stata la calma, dovuta alla lentezza del parlato ed anche all'omogeneità della tonalità del discorso. Paura, rabbia, neutrale e sorpresa sono le emozioni che non avrebbero mai associato a tali registrazioni. Alle volte alla tristezza è stato affiancato il disgusto, in tal caso è stata chiesta la motivazione di tale riscontro emotivo e, per lo più, è stato risposto che quel disgusto che riscontravano, era dovuto ad un atteggiamento annoiato nella spiegazione, come a voler sottolineare la facilità di un argomento trattato e, quindi, un voler dar per scontato che quel ragionamento fosse capito. La noia è un'emozione che molti dei ragazzi hanno aggiunto come opzione durante l'analisi.

Nel caso delle videolezioni di Advanced Design for signal le emozioni escluse sin da subito sono state quelle di gioia, calma, neutralità e rabbia. Al contrario, sono stati identificati con una grande frequenza stati emotivi come paura, tristezza e disgusto. Una volta segnate tali emozioni in tabella, sono state chieste le motivazioni di tale scelta. In particolare, la paura è stata giustificata, correlandola alla difficoltà che il docente mostra nel parlare in lingua inglese. Molti hanno associato tale emozione alla paura di sbagliare e hanno accostato a

questa anche ansia. In più la voce tremula, il continuo inserimento di intercalari nella frase e di versi vari ed il continuo ripetere una parola o parte di una frase prima di concluderla hanno comunicato tale stato emotivo.
La tristezza ed il disgusto, invece, sono stati selezionati in tabella a causa della monotonia della spiegazione; quindi, è stata fatta un'associazione alla cadenza del parlare del professore. In linea generale, il professore non si è ritenuto a suo agio in quella situazione.

Nel caso delle lezioni di Comunicazione multimediale, infine, le emozioni non riscontrate sono tristezza, paura, rabbia, sorpresa. Al contrario tutti hanno riscontrato calma e, in alcuni casi, la gioia. Da un ascolto analitico degli audio osserviamo che, la gioia è associata a battute di scherzo pronunciate dalla docente in alcuni momenti; mentre la calma è sempre rilevata grazie al fatto che le parole risultano distese e mai spezzate ed all'assenza di picchi di intensità più o meno elevati. La professoressa mantiene sempre la stessa tonalità ed intensità per tutta la durata della lezione, ma anche in lezioni condotte in giornate differenti.

Ultimo caso di ricerca è quello rappresentato dalle lezioni del corso di Photonic devices, per queste videolezioni le emozioni riscontrate sono state tristezza e paura, interpretazione che deriva essenzialmente dal tono basso e cupo utilizzato durante la maggior parte delle lezioni dal professore. Solo in alcuni casi veniva associata come ulteriore emozione la gioia, nei pochi casi in cui il professore aumenta il tono di voce e coinvolge maggiormente gli studenti nella sua spiegazione. Questi risultati confermano quanto predetto dalla rete di apprendimento usata su tali videolezioni.

5.3 Conclusioni

Mettendo a confronto i dati predetti dai modelli di apprendimento, presentati nel Capitolo 4, con i dati dedotti sulla base di quanto detto dagli agenti umani, diventa possibile tracciare un filo conduttore ed impostare una conclusione per lo studio affrontato. Nonostante l'incongruenza tra i risultati ottenuti con un modello piuttosto che con un altro, notiamo che tutte le emozioni che vengono tirate fuori dagli audio sono le stesse emozioni riconosciute dai partecipanti dei test di validazione effettuati. Ci sono emozioni che non vengono predette, che sono le stesse che vengono escluse a priori anche dai partecipanti alla ricerca; ed emozioni che vengono prese in considerazioni come opzioni, che vengono tirate fuori anche dai modelli di apprendimento.

Modelli come quello CNN con e senza split, il cui apprendimento viene portato avanti su dataset Ravdess e Tess, che hanno accuracy generale che si aggira negli intorni rispettivamente del 34% e del 54%, colgono meno sfumature emozionali. È possibile vedere che da tutti gli audio questi modelli predicono la stessa emozione, come a cogliere il mood usato dal professore in tutte le lezioni; piuttosto che l'emozione in sé e per sé. Questi si basano sui valori di mfcc estratti, che coinvolgono lo stesso intervallo numerico. Chiedendo ai

partecipanti alla ricerca se pensassero che il professore conducesse o meno le lezioni tutte allo stesso modo, comunicando la stessa propensione all'insegnamento, la risposta risulta sempre positiva. I partecipanti affermano che il mood del professore è riconoscibile e che il suo modo di spiegare è sempre lo stesso, facilmente distinguibile da quello di un altro professore. Inoltre, un'emozione era sempre riconosciuta come principale in tutte le lezioni, per cui il tratto distintivo emozionale può essere definito. Per tale motivo, basandoci su quanto detto, non sono da considerarsi errati i modelli che tirano fuori sempre la stessa emozione da tutti gli audio, come tratto distintivo di quel professore nell'insegnare la sua materia. Modelli, invece, come il CNN con e senza split con Ravdess e Savee, che presentano accuracy intorno rispettivamente al 38% e al 33%, riescono a cogliere più sfumature emozionali, che sono le stesse emozioni che anche gli agenti umani reclutati hanno associato alle registrazioni; ma la scarsa precisione si nota nel momento in cui a parlare è un interlocutore di sesso maschile e questo viene al 50% associato al sesso femminile. Ciò è dovuto al fatto che intercalari detti più volte e, quindi, suoni brevi vengono colti con intensità maggiore. Sappiamo dalla teoria che le donne presentano un'intensità nel loro parlare superiore a quella maschile, ed è per questo motivo che il modello fa confusione e deduce sia una donna, piuttosto che un uomo. Sicuramente, quello che possiamo concludere è che un dataset più vasto e vario, permette di cogliere più sfumature emozionali. Per quest'ultimo motivo si ritiene che con un dataset italiano meglio organizzato e più complesso e completo si possano dare contributi importanti in ricerche, come quella condotta in questa tesi. Nessuna predizione è da ritenersi errata, nessun modello si può affermare tragga fuori delle conclusioni errate. Quello che possiamo affermare è che il problema che ci si è prefissati è un problema complesso sia per l'uomo che per il Machine Learning e che c'è una traccia di realtà, come si è visto, in tutti i modelli presentati. Unica sostanziale differenza è che vengono colte sfumature diverse dai modelli, questo accade per le diversità delle metriche utilizzate, per i dataset utilizzati e per valori di parametri e iperparametri usati diversi.

5.4 Svolgimenti futuri

A conclusione del lavoro di tesi svolto ed illustrato, è stato possibile identificare i modelli di apprendimento automatico per il riconoscimento delle emozioni, a partire da tracce audio, che presentano valori di accuracy e livelli di predizione maggiori rispetto ad altri. Una volta isolati i modelli che fornivano possibilità maggiori in termini di riconoscimento emozionale, è stato possibile effettuare una valutazione delle problematiche a cui si è dovuto far fronte nell'operazione di interpretazione delle emozioni. Fondamentalmente, si è potuta constatare la debolezza del dataset italiano EMOVO e non si è potuto non tenere in considerazione le scarse qualità dell'audio utilizzato e le problematiche derivate dal modo di parlare e condurre la spiegazione da parte dei docenti. Quanto appena detto ha portato come conseguenza ad un'ambiguità nella valutazione dell'emozione da parte dei modelli analizzati.

Ciò che emerge è sicuramente la necessità di incorporare più vocaboli nel dataset su cui viene svolto l'apprendimento della rete, importante sarebbe l'apporto dell'inserimento di intercalari usati frequentemente nel nostro parlare quotidiano. Inserimenti di questo tipo potrebbero portare al riconoscimento della condizione emotiva in cui l'interlocutore si trova e, quindi, facilitare notevolmente l'analisi emozionale. Ovviamente non si sta parlando di dare significato alle parole inserite nel dataset, visto che più volte è stato ribadito che si prescinde da queste, ma sicuramente un inserimento di questo tipo a livello di features, potrebbe essere facilmente riconosciuto dal modello in questione.

Oltre a rafforzare il dataset di riferimento su cui basare la fase di apprendimento, migliorare la qualità degli audio incrementerebbe, sicuramente, i risultati. Quindi, inserire un modo per effettuare questo miglioramento, potrebbe essere un passo di rilievo verso una migliore predizione emotiva.

Sitografia Figure

[1] https://jonathan-hui.medium.com/speech-recognition-feature-extraction-mfcc-plp-455f5a69dd9

[2] https://www.nucleodoconhecimento.com.br/economia-aziendale/apprendimento-profondo

[3] https://medium.com/swlh/an-overview-on-convolutional-neural-networks-ea48e76fb186

[4] https://www.researchgate.net/Figura/A-basic-convolutional-neural-network-structure-for-image-classification-Convolutional_fig1_342316332

[5] https://medium.datadriveninvestor.com/recurrent-neural-network-with-keras-b5b5f6fe5187

[6] https://towardsdatascience.com/lstm-recurrent-neural-networks-how-to-teach-a-network-to-remember-the-past-55e54c2ff22e

[7] https://www.domsoria.com/2021/07/cosa-sono-le-curve-auc-roc/

[8] https://www.google.com/url?q=https://medium.com/swlh/machine-learning-how-to-prevent-overfitting-fdf759cc00a9&sa=D&source=docs&ust=1657683639952759&usg=AOvVaw0I-fT9bqZcNnYIpvZTlAsX

[9] https://github.com/MITESHPUTHRANNEU/Speech-Emotion-Analyzer

[10]https://www.andreaminini.com/ai/machine-learning/overfitting

[11] https://amueller.github.io/aml/04-model-evaluation/1-data-splitting-strategies.html

[12] http://court-st-etienne.be/jgss.asp?iid=382224936&cid=66

[13] https://www.google.com/url?q=https://it.mathworks.com/discovery/support-vector-machine.html&sa=D&source=docs&ust=1657683639951915&usg=AOvVaw07oxccoZNYPykyYNATcpYF

[14] https://ja.d2l.ai/chapter_deep-learning-basics/mlp.html

Bibliografia

[1] ***A Proposal for Multimodal Emotion Recognition Using Aural Transformers and Action Units on RAVDESS Dataset*** di Cristina Luna-Jiménez , Ricardo Kleinlein, David Griol, Zoraida Callejas, Juan M. Montero and Fernando Fernández-Martínez, 2002.

[2] ***CNN+LSTM Architecture for Speech Emotion Recognition with Data Augmentation -*** Caroline Etienne, Guillaume Fidanza, Andrei Petrovskii, Laurence Devillers, Benoit Schmauch

[3] ***Combining convolutional neural networks for emotion recognition*** - Christina Huang

[4] ***A Comprehensive Review of Speech Emotion Recognition Systems -*** Taiba Majid Wani; Teddy Surya Gunawan; Syed Asif Ahmad Qadri; Mira Kartiwi; Eliathamby Ambikairajah

[5] ***Deep Learning Techniques for Speech Emotion Recognition, from Databases to Models*** Babak Joze Abbaschian, Daniel Sierra-Sosa and Adel Elmaghraby

[6] ***Shallow over Deep Neural Networks: A empirical analysis for human emotion classification using audio data -*** Chandresh S. Kanani, Karanjit Singh Gill, Sourajit Behera, Anurag Choubey, Rohit Kumar Gupta, and Rajiv Misra ·

[7] Noam Amir. ***Classifying emotions in speech: a comparison of methods***, 2001.

[8] ***CNN+LSTM Architecture for Speech Emotion Recognition with Data Augmentation-*** Caroline Etienne, Guillaume Fidanza, Andrei Petrovskii, Laurence Devillers, Benoit Schmauch, 2018

[9] ***Academic Emotion Classification and Recognition Method for Large-scale Online Learning Environment—Based on A-CNN and LSTM-ATT Deep Learning Pipeline Method*** Xiang Feng, Yaojia Wei, Xianglin Pan, Longhui Qiu and Yongmei Ma

[10] ***Speech Emotion Recognition of Teachers in Classroom Teaching*** Liang Jie, Zhao Xiaoyan, Zhang Zhaohui

[11] ***Multimodal Emotion Recognition on RAVDESS Dataset Using Transfer Learning*** ***Sensors*** 2021 · Cristina Luna-Jiménez, David Griol, Zoraida Callejas, Ricardo Kleinlein, Juan M. Montero, Fernando Fernández-Martínez ·

[12] ***EMOVO Corpus: an Italian Emotional Speech Database*** Giovanni Costantini, Iacopo Iadarola, Andrea Paoloni, Massimiliano Todisco

[13] ***Speech emotion recognition with deep convolutional neural networks -*** Dias Issa, M.Fatih Demirci, AdnanYazici

[14] ***Speech emotion recognition using deep 1D & 2D CNN LSTM networks -*** lJianfeng Zhaoab, XiaMaoa, LijiangChen

[15] ***Deep Learning for Emotion Recognition on Small Datasets using Transfer Learning -*** Hong-Wei Ng, Viet Dung Nguyen, Vassilios Vonikakis, Stefan Winkler

[16] ***Audio and face video emotion recognition in the wild using deep neural networks and small datasets -*** Wan Ding, Mingyu Xu, Dongyan Huang, Weisi Lin, Minghui Dong, Xinguo Yu, Haizhou Li

[17] ***Detection of Emotion of Speech for RAVDESS Audio Using Hybrid Convolution Neural Network*** - Tanvi Puri, Mukesh Soni, Gaurav Dhiman, Osamah Ibrahim Khalaf, Malik alazzam, Ihtiram Raza Khan
[18] ***Emotion Recognition from Speech*** **-** Kannan Venkataramanan, Haresh Rengaraj Rajamohan
[19] ***Automatic Speech Emotion Recognition Using Machine Learning*** **-** Leila Kerkeni, Youssef Serrestou, Mohamed Mbarki, Kosai Raoof, Mohamed Ali Mahjoub and Catherine Cleder
[20] ***Speech emotion recognition using convolutional and Recurrent Neural Networks*** **-** Wootaek Lim; Daeyoung Jang; Taejin Lee
[21] ***Applying Machine Learning Techniques for Speech Emotion Recognition*** **-** K. Tarunika; R.B Pradeeba; P. Aruna
[22] ***Speech emotion recognition with deep learning*** **-** Pavol Harár; Radim Burget; Malay Kishore Dutta

Ringraziamenti

Quando mi sono iscritta all'università non avevo la minima idea di chi volessi diventare e di dove il percorso scelto mi avrebbe potuto portare. Ho iniziato e concluso la mia carriera universitaria accompagnata da mille paure, compresa quella di non aver preso le giuste scelte, e dall'ansia di non essere all'altezza. Adesso, mi ritrovo alla fine di un percorso che mi ha reso quella che sono, che mi ha fatto crescere, e che mi ha regalato grandi soddisfazioni. Auguro a chiunque di intraprendere un percorso come il mio, fatto di alti e bassi, ma da cui possa scaturire sempre una grande ammirazione per se stessi. Auguro a chiunque di provare la gioia di una riuscita, ma anche la tristezza di un fallimento. Ringrazio me stessa per averci sempre creduto e per essere sempre andata avanti nonostante tutto. Ringrazio chi ci ha creduto insieme a me, chi c'è sempre stato e chi mi ha sostenuto durante l'intero percorso; grazie per aver condiviso con me le esperienze più importanti. Auguro a chiunque intraprenda un percorso di qualsiasi tipo di trovare persone speciali come è successo a me; grazie agli amici conosciuti in questa splendida città, alle mie "coinquiline", che sempre rimarranno tali perché non ci unisce più lo stesso tetto ma qualcosa di estremamente più importante, e agli amici che sono riusciti a trasformare Villa in Casa. Un ringraziamento speciale va a Noemi per avermi sopportato e per essere stata al mio fianco nonostante io l'abbia reso difficile; riconosco di essere stata insopportabile. Un grazie va a Leonardo da cui, nonostante i due anni passati insieme, non ho mai avuto una parola di conforto; ammetto, però, che l'allegria e la leggerezza che lo contraddistinguono sono stati fondamentali. Entrambi sono sempre riusciti a strapparmi un sorriso nonostante le difficoltà, entrambi hanno creduto in me anche quando io non ero in grado di farlo. Un grazie va anche a chi non mi ha capita, a chi ha scelto di non rimanere al mio fianco fino alla fine, perchè è grazie a loro che sono cresciuta e ho imparato cosa significa bastarsi sempre. Vorrei ringraziare, infine, le persone a me più care: gli amici di una vita e la mia famiglia, per essere stati sempre presenti e pronti ad aiutarmi. Un grande ringraziamento va a mia madre e mio padre che mi hanno sostenuto, sia moralmente che economicamente, e mi hanno permesso di arrivare a questo importante risultato. Sperando che questo sia punto di arrivo e contemporaneamente di partenza della mia vita, li ringrazio per essere sempre stati un punto di riferimento. Ringrazio mio fratello, la persona più importante della mia vita, per essere stato la distrazione di cui avevo bisogno, e soprattutto gli sono grata per essere riuscito a farmi ridere anche quando non ne avevo voglia.

Spendo, per concludere, delle parole per chi sopraffatto da pressioni, ansie e paure, si è arreso e non è arrivato fino in fondo. Dedico questa tesi a chi ha pianto notti intere, a chi non riusciva a respirare per l'ansia, a chi si è dato solo colpe e a chi più di una volta si è chiesto se ne valesse la pena.

Dedico questa tesi a chi non si è sentito abbastanza o all'altezza. Nessuno è l'opinione di uno sconosciuto. Siamo tante cose, ma di sicuro non siamo quel fallimento che qualcuno ci fa credere di essere.